CAPITAL UNIVERSITY OF SCIENCE AND
TECHNOLOGY, ISLAMABAD

Evaluation and Correction of Inconsistency and Incompleteness in DBpedia Knowledge Graph

by

Mubashar Bilal

A thesis submitted in partial fulfillment for the
degree of Master of Science

in the

Faculty of Computing
Department of Computer Science

2023

I dedicate this work to my respectful teachers and my parents who helped me to accomplish this work. Especially to my parents because without their prayers, moral support and best wishes i couldn't accomplish this work.

CERTIFICATE OF APPROVAL

Evaluation and Correction of Inconsistency and Incompleteness in DBpedia Knowledge Graph

by

Mubashar Bilal

(MCS193038)

THESIS EXAMINING COMMITTEE

S. No.	Examiner	Name	Organization
(a)	External Examiner	Dr. Munir Ahmad	BIIT, Rawalpindi
(b)	Internal Examiner	Dr. Mohammad Masroor Ahmed	CUST, Islamabad
(c)	Supervisor	Dr. Muhammad Abdul Qadir	CUST, Islamabad

Dr. Muhammad Abdul Qadir
Thesis Supervisor
January, 2023

Dr. Abdul Basit Siddiqui
Head
Dept. of Computer Science
January, 2023

Dr. Muhammad Abdul Qadir
Dean
Faculty of Computing
January , 2023

Author's Declaration

I, **Mubashar Bilal** hereby state that my MS thesis titled "**Evaluation and Correction of Inconsistency and Incompleteness in DBpedia Knowledge Graph**" is my own work and has not been submitted previously by me for taking any degree from Capital University of Science and Technology, Islamabad or anywhere else in the country/abroad.

At any time if my statement is found to be incorrect even after my graduation, the University has the right to revoke my MS Degree.

(Mubashar Bilal)
Registration No: MCS193038

Plagiarism Undertaking

I solemnly declare that research work presented in this thesis titled "**Evaluation and Correction of Inconsistency and Incompleteness in DBpedia Knowledge Graph**" is solely my research work with no significant contribution from any other person. Small contribution/help wherever taken has been duly acknowledged and complete thesis has been written by me.

I understand the zero tolerance policy of the HEC and Capital University of Science and Technology towards plagiarism. Therefore, I as an author of the above titled thesis declare that no portion of my thesis has been plagiarized and any material used as reference is properly referred/cited.

I undertake that if I am found guilty of any formal plagiarism in the above titled thesis even after award of MS Degree, the University reserves the right to withdraw/revoke my MS degree and HEC and the University have the right to publish my name on the HEC/University website on which names of students are placed who submitted plagiarized work.

(Mubashar Bilal)
Registration No: MCS193038

Acknowledgement

Thanks to Allah Almighty for blessing me with wisdom and strength to complete the dissertation. Being an MS graduate at Capital University of Science and Technology has been a magnificent and challenging experience. During the degree, I have found clear guidelines in shaping my academic career. Here is a humble tribute to all those people. I would like to express my sincerest appreciation to my enthusiastic supervisor, **Dr. Muhammad Abdul Qadir** for his supervision, assistance, and immense knowledge. I am sincerely thankful to him for his constant support, motivation, and patience. His invaluable help of constructive comments and suggestions throughout the thesis work has contributed to the success of this research. It has been an amazing experience and I thank him wholeheartedly for his tremendous support. I would like to thank my family who had motivated me continuously to achieve this milestone. A word of applause for my friends and classmates who had assisted me in sharing knowledge and other resources required to conduct research. Thank you all.

(Mubashar Bilal)

Abstract

A knowledge graph structures domain specific data (concepts and relations) in the form of a graph which can be meaningfully interpreted by machines to generate relevant answers for queries. DBpedia is one of the popular knowledge graph extracted from Wikipedia. Evaluation of a knowledge graph for inconsistent or incomplete knowledge in order to get precise and correct result of the queries, is crucial for a reliable knowledge graph. This thesis evaluates DBpedia for inconsistency and incompleteness. These errors could be due to one of the three possible reasons. First, the errors in the source data (Wikipedia) that may be due to negligence or less knowledge provided in the Wikipedia. Second, the errors in the conversion process which could either be due to DBpedia mappings or some error in the DBpedia extraction logic, and lastly, errors in the DBpedia ontology which might be due to wrong hierarchy of classes (concepts), properties (relations) or incorrect domain or range definitions etc. It is demonstrated that how these errors can be discovered and corrected by thorough investigation of DBpedia knowledge graph in four phases, that is discovery phase, analysis phase, correction phase and evaluation phase. More than 5300 inconsistent and incomplete instances were discovered in the knowledge graph in the discovery phase. Another activity in the discovery phase is the evaluation of DBpedia ontology for incorrect classes or relations. Incorrect domain and range definitions of many relations and incorrect hierarchy of the concepts were discovered. After analysis and correction of these errors, same investigation process is used for the evaluation. Finally results are compared with the previous errors from the discovery phase. 98% of these errors were removed by making changes in the DBpedia extraction framework, ontology and mappings. Twelve new concepts and fifteen mappings were introduced in the DBpedia ontology to resolve the incompleteness. DBpedia ontology was also analyzed for inconsistency and incompleteness. Four inconsistencies in relations and two inconsistent hierarchies were discovered in ontology, which were also corrected. All the corrections which were made in this thesis were shared with DBpedia community, they agreed and accepted the changes with appreciation.

Contents

List of Figures

List of Tables

Abbreviations

AST	Abstract Syntax Tree
Dbo	DBpedia Ontology
DBP	DBpedia
Dbp	DBpedia Property
DSR	Design Science Research
KG	Knowledge Graph
ORE	Ontology Repair and Enrichment
OWL	Web Ontology Language
QAKIS	Question Answering WikiFramework-based System
RDF	Resource Description Framework
Regex	Regular Expression
SPARQL	Simple Protocol and RDF Query Language
UMBEL	Upper Mapping and Binding Exchange Layer

Chapter 1

Introduction

1.1 Knowledge Graph

Semantically annotated structured data as an instance of ontology is known as Knowledge Graph. It is used to store interlinked entities. The domain data in the knowledge graph follows a domain definition in the form of ontology to enable us to intelligently query the data [1]. It stores information usually in resource description framework (RDF) triple format comprising of concepts and relationship between them as per the domain structure. A knowledge graph is a special kind of database which stores knowledge in a machine readable form and provides a means for information to be collected, organized, shared, searched and utilized. Queries like all cities with low criminality, warm weather and open jobs can be answered by using these knowledge graphs.

Ontology(usually comprises of domain concepts in form of classes and relations between them) provides the required structure to knowledge graph, therefore a slight change in the ontology can cause the knowledge graph to regenerate itself. This can be advantageous when we want to correct certain relations between concepts as per domain knowledge. We only need to correct the relevant ontology classes and properties and all the related instances which might be thousands in number will automatically be corrected by the reasoning system of the knowledge graph. There are variety of applications and use cases of knowledge graph which includes journalism, entertainment, and pharmaceuticals [2] etc. There are many

publicly available knowledge graphs which performs very well in answering relevant queries. These knowledge graphs are mostly populated from semi-structured or unstructured data by extracting data from the source, annotating it with domain specific concepts or properties and storing it as RDF triples. Among them are DBpedia, Yago and Wikidata etc [2].

1.1.1 Difference Between Knowledge Graph and Database

A knowledge graph is a graph-based data model that represents real-world entities and the relationships between them. It is different from a traditional database, which stores data in tables with predefined schemas and is optimized for fast data retrieval and updates. Knowledge graphs, on the other hand, are designed to represent complex and interconnected domain specific data and to support querying and reasoning over the data. They use a domain specific schema that allows for the representation of arbitrary relationships between entities and the incorporation of external knowledge sources.[2]

1.1.2 Knowledge Graph Process and Characteristics

Figure 1.1 illustrates the creation of knowledge graph process in which a converter is used to annotate unstructured or semi-structured data into a semantic web data which comprises of a domain ontology, instances and URIs.

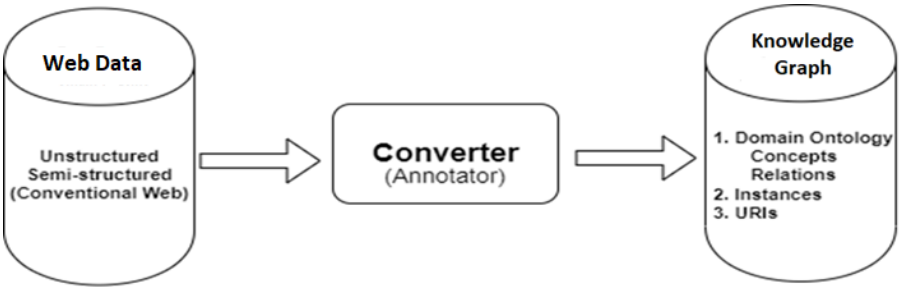

FIGURE 1.1: Knowledge graph process

A good knowledge graph complies with following three characteristics [1, 2]:

1. **Completeness:** It should contain all the information about a domain that is supposed to be in it or can be inferred from the available concepts and relations.
   A KG which doesn't have all the concepts about a domain which is meant to be in it is considered **incomplete**.

2. **Consistency:** It should be free from ambiguous or contradictory statements to define concepts and relations about a domain.
   A KG which has statements that are ambiguous is considered **inconsistent**.

3. **Conciseness:** It should not be redundant and shouldn't contain statements which can be generated from already given information, for-example age can be calculated from birth date therefore only birth date is sufficient.
   A KG which contains information that can be deduced from existing information is considered **redundant**.

A large amount of data is being generated each day on the web which could be very useful but the problem with conventional web is that it often does not provide relevant answers to query. Data becomes a big dump if we can't find relevant answers. Knowledge graph solves this problem by storing the unstructured or semi-structured data in a semantically annotated structured form.

1.2 DBpedia

The DBpedia project was started in 2007 by the University of Leipzig and University of Mannheim. DBpedia is a Knowledge Graph which extracts data mainly from Wikipedia info boxes and populates it as per the very simple and shallow multi-domain ontology [3]. DBpedia is a widely used knowledge graph in research because:

1. It is not restricted to a particular domain and mostly targets Wikipedia info boxes, which are publicly available [2].
2. It is available in multiple languages [3].
3. Updated continuously to incorporate changes in the source data [3].
4. Both the source data and Knowledge graph are publicly accessible therefore many improvements can be made by analyzing the source data and knowledge graph (KG).

The DBpedia ontology contains 768 concepts described by 3000 relations and total instances are more than four million [4].

TABLE 1.1: DBpedia instances [4]

Instance Type	**Count**
Resource (overall)	4,828,418
Place	967,491
Person	1,592,912
Work	552,115
Species	190,369
Organization	317,867
Other	1,207,664

Figure 1.2 shows the conversion of data from Wikipedia schema to DBpedia ontology. DBpedia extracts data from Wikipedia and store it in RDF form.Then different queries can be performed on DBpedia to get relevant answers.

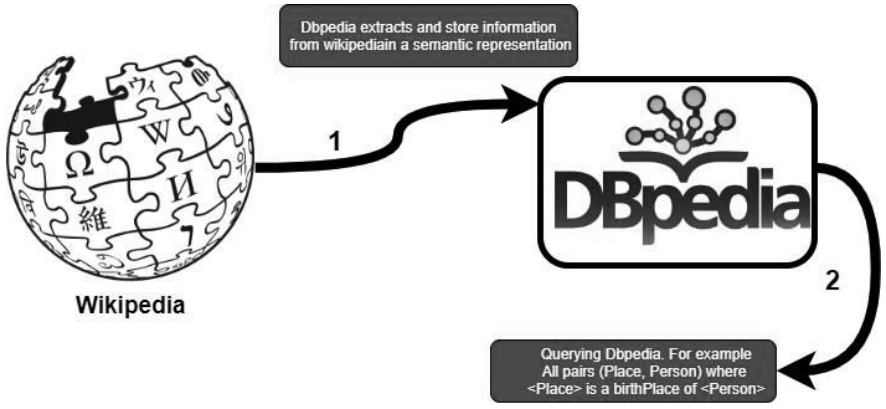

FIGURE 1.2: Wikipedia to DBpedia process[5]

1.3 Structure of DBpedia

DBpedia comprises of three main components [4]:

1.3.1 Ontology:

Ontology defines all the concepts (termed as classes) and relations (termed as properties) about a domain. The classes and properties are created carefully by the DBpedia research community. As everything is open source and accessible publicly, it can be updated by the community.

1.3.2 Mappings:

Mappings transforms the information from source namespace to destination namespace, in DBpedia knowledge graph from Wikipedia schema (namespace) to DBpedia ontology (namespace). Wikipedia is a semi-structured database that contains key-value pairs for an entity represented in the info box [3]. DBpedia defines mappings that tell which Wikipedia info box key is mapped to which DBpedia property

whereas type of info box is mapped to DBpedia class, for example Wikipedia Person info box is mapped to DBpedia Person class.

1.3.3 Extraction Framework:

Extraction framework is the tool which populates DBpedia knowledge graph by extracting data from Wikipedia, considering all the classes and the mappings available. It consists of following core components [6].

1. **Source:** It contains Wikipedia pages.
2. **WikiParser:** Parses each page and convert it into Abstract Syntax Tree (AST).
3. **Extractor:** It maps the page nodes to DBpedia classes and properties using the mappings and ontology definitions.
4. **Destination:** It publishes the mapped data in the form of RDF triples.

FIGURE 1.3: DBpedia extraction framework working

DBpedia extraction framework comprises of different extractors for different purpose [6] like PageIdExtractor, InfoboxExtractor, and WikilinkExtractor etc.

1.4 Errors in DBpedia

There could be three types of errors in DBpedia, that is, syntax, logical and semantic errors. Syntax errors can be discovered by RDF validators, logical errors by Reasoners like Pellet, fact++ but semantic errors cannot be discovered by such tools because they are both syntactically and logically correct and real world

knowledge is required for their discovery [7]. DBpedia knowledge graph contains different semantic errors such as inconsistency, incompleteness and redundancy.

1.4.1 Inconsistency

Table 1.2 shows DBpedia profile of entity **Karen McCarron**. There exists an inconsistency of having more than one birth date which could not be true in real life for a **Person**.

TABLE 1.2: DBpedia profile of Karen McCarron [8]

DBpedia page	**Karen McCCarron**
Type	Person
Abstract	Karen Frank-McCarron (born December 20, 1968) is a German-born American pathologist convicted in Illinois of first degree murder of her autistic daughter Katherine "Katie" McCarron.
Birth name	Katherine Marie McCarron
Birth date	1968-12-20 2002-07-22
Birth place	Peoria,_Illinois West$_G ermany$

1.4.2 Incompleteness

There exist Wikipedia info boxes against which there are no DBpedia classes to mapped to. One such info box is **Infobox Academic**. Table 1.3 shows that this info box contains many important properties like Education, Thesis etc. The **Infobox Academic** contains more than 9000 instances but not mapped to DBpedia due to incomplete definitions of concepts and relations.

TABLE 1.3: Wikipedia infobox academic template [9]

Infobox	**Academic**
name	Academic name
education	Education level
thesis_title	Complete thesis title
thesis_year	Year in which thesis was completed
doctoral_advisor	Name of doctoral advisor

1.4.3 Redundancy

Birth year and birth date are two separate properties in DBpedia ontology, although birth date is sufficient.

DBpedia contains issues due to problems in any of the three main components, Ontology, Mappings and/or Extraction Framework. A small issue in these components may result in a large amount of incorrect data populated in a knowledge

graph. Aim of this study is to find those errors in DBpedia, investigate the possible cause of these errors and then remove the errors by making changes in the system as required.
A knowledge graph is built to answer relevant and meaningful queries precisely. If the underlying knowledge graph contains errors, the primary objective of answering the relevant queries cannot be met. Therefore, it is very important to identify and remove errors from a knowledge graph. Most of the work is done on incompleteness and inconsistency therefore redundancy is not considered in this study.

1.5 Research Gap

DBpedia suffers from Incompleteness and Inconsistency problems [2]. The errors that can cause incorrect and irrelevant results for query must be removed in order to make it a true semantic web to provide relevant answers.
Research gap is identified by carefully evaluating the literature and with the help of thorough experiments on DBpedia. See section 2.3

1.6 Research Questions

Following are the research questions which need to be answered in order to fill the research gap.

1. **What are the reasons of inconsistency and incompleteness in DBpedia?**

   There can be different reasons behind inconsistency and incompleteness like error in source data or error in conversion process. It is required to discover the source of error(component of DBpedia system responsible for the error).

Section 3.2.1.9 and section 3.2.2.2 shows the reasons behind inconsistency problems and section 3.2.3.2 shows the reason of incompleteness problem.

2. **How to remove these errors in order to have consistent and complete semantic web?**

   After discovering errors, it is being needed to eliminate the errors by modifying either source data or DBpedia components. Different solutions and ways must be proposed and implemented in order to make DBpedia error free.
   Section 3.2.1.9, 3.2.2.3 and 3.2.3.3 shows the solution of inconsistency and incompleteness problems.

3. **What are the missing concepts and relations in DBpedia ontology and how to enrich it?**

   Incomplete definitions of domain are often cause of semantic errors in DBpedia, there is a need to discover missing concepts which exist in source data and then find a way to add them in DBpedia.
   Section 3.3.2 shows the missing concepts and relations in DBpedia and section 3.3.3 shows how to add missing concepts and relations in DBpedia.

1.7 Scope of Research

The scope of this research is to discover, analyze and remove inconsistency and incompleteness from DBpedia knowledge graph. The errors will be discovered using semantically enriched complex queries and analysis of DBpedia ontology. All the errors will be analyzed by comparing the information in the source data, looking at the problems in the extraction framework and DBpedia ontology. Then these errors will be removed by correcting the source of error.

1.8 Methodology

This study employs design science research (DSR) methodology[10], as illustrated in Figure 1.4. The first step in this process involves identifying the problem at hand and formulating research questions. To gain a deeper understanding of the problem, we review the existing literature and identify any existing techniques and the challenges they present. DSR aims to produce high-quality, valuable, and acceptable research that can be published in academic outlets. To discover and correct different errors in DBpedia knowledge graph, different experiments have been performed and after correction, results are evaluated using same discovery process.

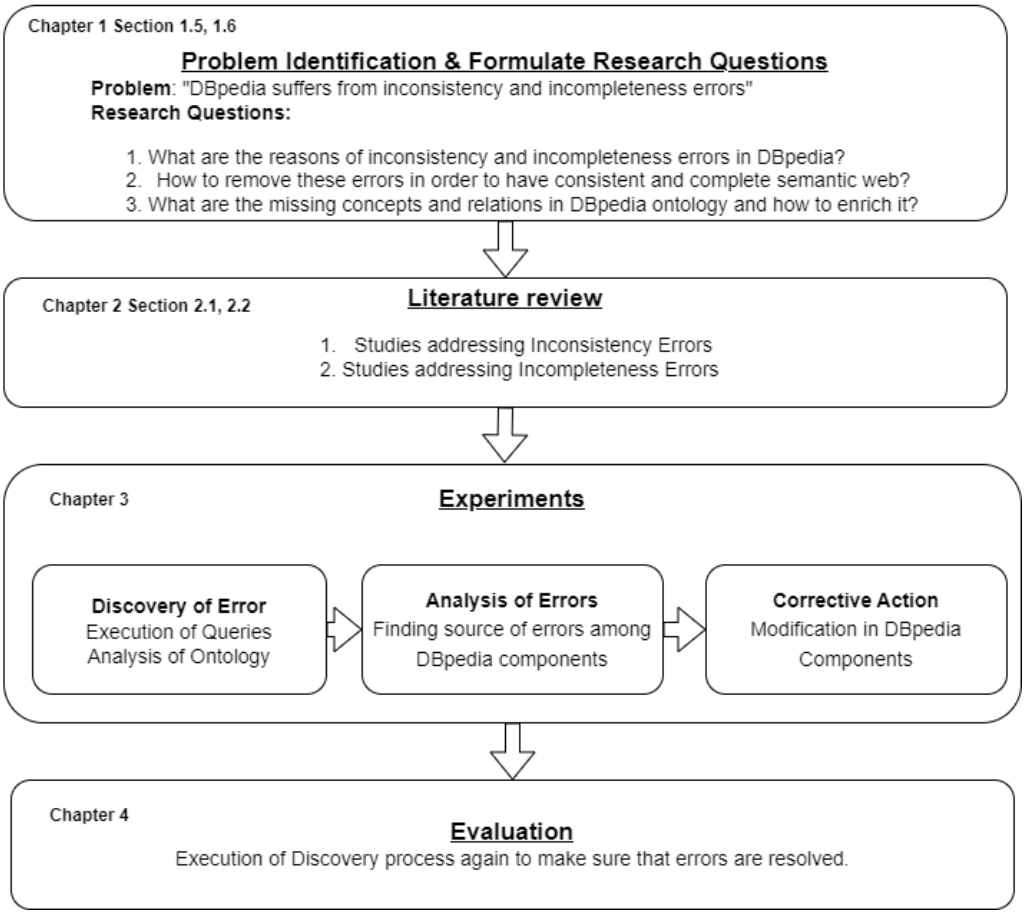

FIGURE 1.4: Methodology diagram

Summary

This chapter explained DBpedia and answered few questions about it like what is

DBpedia and different components of DBpedia? What are the issues in DBpedia? How these issues can be resolved?. This chapter also discussed the research gap, research questions, scope of research and methodology to find and resolve issues.

Chapter 2

Literature Review

This chapter presents literature review and highlights some key issues which led to the proposed approach. This chapter evaluates studies addressing inconsistency and incompleteness problems. Based on previous researches literature is categorized into two sections. Section one addresses studies related to inconsistency problem and section two addresses studies related to incompleteness problem.

2.1 Discovering and Removing Inconsistencies

There are many studies that are addressing the problem of inconsistencies in DBpedia. These inconsistencies can exist both in ontology and instances. Studies [7, 11–13] focused on discovering inconsistencies using different techniques only and no solution to inconsistencies were provided, however studies [14–19] along with discovery also provided solutions to the discovered inconsistencies.

Sack H. et al. [7] mentioned that there exists three types of errors in knowledge graph Syntax errors, Logical errors and Semantic errors. Syntax errors can be detected by RDF parser, logical errors can be detected by using some reasoner like fact++. However semantic errors are very difficult to detect because they require real world knowledge. To overcome this difficulty author presented an approach for the detection of semantic errors in the dbpedia knowledge graph by transforming the semantic errors into logical errors which can then be detected by some reasoner. They enriched the ontology with domain, range and disjoint

axioms so that contradicting logical statements arise which will be detected.

Disjointness axioms help in consistency checking and evaluation of ontologies. Disjointness modeling is challenging and time taking. This study [11] created an approach which helps to enrich learned or manual ontologies with the help of disjointness axioms. Disjointness axioms were learned using association rule mining technique. This approach helps to increase richness, usefulness and quality of ontology. This approach behave like a human annotators. Highest accuracy of 90.9% was achieved by this model. For disjointness modeling lessons from human annotators were learned.

Yanfang Ma T. et al. [12] extends the existing study [11] of association rule mining using Apriori algorithm to discover inconsistencies in the DBpedia knowledge graph. They found five problems in the existing approach. First problem was that classes with no instances could not be included in the transaction table, second problem was that incorrect rules of child class disjoint with parent class were generated, third problem was that incorrect rules of parent class disjoint with child class were generated, fourth problem was that contradictory rules were generated which stated that class A is subclass of class B but class B is also disjoint with class A, fifth problem was that for many of the disjointness rules no symmetric rules were generated. After discovering the problems, they came to conclusion that ignoring class hierarchy during rule mining was one of the most important reason for the mentioned five problems. A modified existing technique was implemented considering class hierarchy and then using these rules, they detected inconsistencies in DBpedia and zhishi.me. They worked on detection of inconsistencies, however investigation of reason behind inconsistencies and removal of inconsistencies is not being done in the research.

Wikipedia info box keys are mapped to corresponding DBpedia classes and this is mostly done by community. Due to language and knowledge differences these mappings are not consistent. One can see the mappings manually to know the inconsistencies. Kontokostas D. et al. [13] presented a machine learning approach

for automatic detection of incorrect mappings. They utilized different language datasets and then defined some rules to filter out the incorrect mappings by taking advantage of overlapping of different languages dataset. They achieved highest performance for Multilayer Perceptron (94.25%). The paper provides a generic and automatic approach as compared to domain specific approach, however manual annotation of different languages datasets is required for training and testing purposes.

Hervieux N. et al [14] proposed the supervised method to detect incorrect type assignment to entities and also predict type for unseen entities. They converted entities into vectors, Wikipedia entities into word2vec and DBpedia entities recourse2vec. These vectors are used as a feature set for machine learning classifiers that detect if the type assigned to an entity is incorrect. The intuition behind this approach is that vectors of entities of the same type will be closer to one another in an n-dimensional continuous vector space than vectors of entities of different types and most entities of the same type share the same properties. Their results are promising and effective as they used external data of Wikipedia to reduce error. Three classifiers (Nearest Centroid, kNN, Na¨ıve Bayes) are used in the research and their best performing classifier is Nearest Centroid with accuracy of 97%. They also predicted types of unseen entities.

Villata S. et al. [15] explained that when a same query is given to different DBpedia chapters, they might result in conflicting results or one might subsume others, therefore author proposed an automatic framework for the detection of such inconsistencies in the answers returned by a query endpoint. They discovered semantic relations between different answers. They implemented bipolar fuzzy labeling algorithm to assign the acceptance score. They utilized QAKIS (Question Answering WikiFramework-based System) for the implementation and evaluation of their approach and they achieved highest F-measure of 0.97.

N.A Khan [16] focused on the family relations and manually found number of inconsistencies in Wikipedia and DBpedia related to family relations and removed

them. Author wrote a number of queries which retrieved inconsistent results and then investigated the reasons behind these inconsistencies and removed those from Wikipedia and DBpedia. Author extended DBpedia ontology with new family classes and corrected domain and range of few properties. This resulted in removal of a lot of inconsistent results. Wikipedia profiles of different entities were manually corrected to remove inconsistencies.

Sheng et al. [17] extended the DBpedia ontology with UMBEL(Upper Mapping and Binding Exchange layer) which is a subset of Opencyc knowledge graph to obtain disjoint axioms for dbpedia for discovery of inconsistencies in DBpedia. Author used a MapReduce model to discover five different types of Inconsistencies in DBpedia. Those inconsistencies are undefined classes/properties, violation of range of data type properties by literals, one class is subclass of other class and at the same time disjoint with that class, class is a subclass of two disjoint classes and entity definition of members of disjoint class is not valid. Author also analyzed each type of inconsistency and then proposed a solution for each of the five inconsistencies.

DBpedia uses huge data from Wikipedia. DBpedia is used by many projects to depict and relate different objects. Wikipedia's content is growing with the passage of time. As DBpedia gets data from Wikipedia, it must be updated continuously. DBpedia is either updated through periodically altered large dumps of data or through a software known as DBpedia Live which update RDF triples. When data is in different languages then it can create inconsistencies during extraction. In study [18], inconsistent classes set was defined. Then the frequency of occurrence of these inconsistencies were observed. This approach tried to figure out inconsistencies (which were in languages other than English) at the time of extraction. Author proposed a mechanism to correct inconsistencies in DBpedia live. Two types of inconsistencies were addressed which are Domain Violation and Range Violation. By using this mechanism Range inconsistencies were dropped to 17.35% from 33.58% and Domain inconsistencies dropped to 7.62% from 13.38%. Effectiveness was still limited because version of resource in foreign language didn't

had its equivalent in English Language.

DBpedia is the knowledge graph which extract data from Wikipedia. It depicts and relate different objects. But it is not error free. Range Violation error is the major error in DBpedia knowledge base. The study [19] discussed the range violation error and proposed the method to remove these errors. This error occurs when value of the object is not in the required range of triples. For example, the triple <dbr :Sedo, dbo : locationCountry, dbr : Cologne> is erroneous because Cologne is city but predicate required object which is of type Country. Because of Range Violation errors, information in Dbpedia is erroneous and it also effects other application which are using Dbpedia as a data source. Because of Human error, Wikipedia info boxes also contain inconsistencies. This method finds the inconsistent objects from reduced search space and replace with consistent objects. To calculate the score of candidate objects, two methods are used which are Graph Method and Keyword method. Table 2.1 shows the contribution, strength and weakness of existing studies related to discovering and removing inconsistencies.

TABLE 2.1: Analysis of studies focused on inconsistencies

Contribution	**Strength**	**Weakness**
[7] Axioms like properties, range and domain restriction has been added to improve DBpedia Ontology.	They enriched the ontology and detected inconsistencies using enriched ontology as well.	Their primary focus was on discovering errors and didn't provide an approach to resolve errors.
[11] Use of correlation analysis, and Negative Association Rule Mining for learning disjointness to enrich knowledge repositories.	They discovered disjointness among different classes and added the disjointness axioms to DBpedia ontology for detecting inconsistency.	They did not propose an approach to correct the errors.

Contribution	Strength	Weakness
[12] Detection of inconsistencies occured due to ignoring class hierarchy during rule mining.	They improved the previous study of inconsistency detection using association rule mining.	They did not focus on discovering incompleteness problem and providing solution for inconsistencies
[13] Presented a machine learning approach for automatic detection of incorrect mappings. They utilized different language datasets and then defined some rules to filter out the incorrect mappings by taking advantage of overlapping of different languages dataset.	One of the few papers which targeted mappings for inconsistency detection, many of the errors are resolved without modifying Wikipedia schema or DBpedia ontology.	The proposed approach didn't provide a scheme to modify ontology or extraction framework.
[14] Detect assignment of erroneous type by converting entities into vectors and by comparing similarities between them.	One of few papers which discovered errors in the extraction process and also provided solution.	They did not consider extraction framework as the source of error.
[15] Proposed an automatic framework for the detection of inconsistencies occured due to conflicted answers returned by a query endpoint.	A more close approach to detect inconsistent and incomplete instances with real world knowledge from other endpoints.	They did not provide solution if all the answers provided by all endpoints are wrong or less answers are correct.

Contribution	Strength	Weakness
[16] Focused on the family relations and manually found number of inconsistencies in Wikipedia and DBpedia related to family relations and removed them by extending DBpedia ontology with new family classes and by changing domain and range of few properties.	They worked both on inconsistency and incompleteness errors with human knowledge about a domain.	They did not correct errors from extraction framework.
[17] Extended the DBpedia ontology with UMBEL (Upper Mapping and Binding Exchange layer) which is a subset of Opencyc knowledge graph to obtain disjoint axioms for dbpedia for discovery of inconsistencies in DBpedia. Author used a MapReduce model to discover Inconsistencies in DBpedia.	They used a more concrete and restricted external ontology to enrich DBpedia ontology.	The errors of external ontology are also inherited and didnt focus on other two core components of DBpedia.
[18] Focused on range and domain violations to make language chapters other than English language more suitable.	A much improved chapter of DBpedia is used as a basis to improve other chapters as well.	They did not focus on resolving hierarchical errors in DBpedia ontology.

Contribution	Strength	Weakness
[19] Discussed the range violation error and proposed the method to remove these errors by finding the inconsistent objects from reduced search space and replace with consistent object.	Along with discovering errors, they also proposed a suitable approach for removing these kind of errors they discovered.	DBpedia ontology, mappings and extraction framework are not considered as the source of errors.

2.2 Knowledge Graph Completion

This section evaluates studies addressing the incompleteness problem. The incompleteness problem can also exist at both levels, ontology as well as instances.

Count and size of semantic knowledge web bases is increasing rapidly. It is difficult to maintain these knowledge bases. The study [20] provides the tool named ORE (Ontology repairing and enrichment) to repair and enrich the ontologies. This tool detects the problem in knowledge bases and provide guidance to repair it. ORE helps to extend the ontology by using supervised machine learning approach. Repositories (Protégé[21] and TONES) are used for testing purpose. Interface is required for ORE to use it for analysis of knowledge bases.

Along with info boxes, wikipedia also contains useful information in article text. However, this Information is not very much utilized in DBpedia Knowledge Graph. Author proposed [22] an automated system to extract useful information from text using semantic parser and coreference solver and map this information to DBpedia namespace in the form of rdf triples. In first step Wikipedia markups filtering is performed and unstructured text is provided to next step. Then the links are extracted and semantic parsing of text is performed. Coreference resolution detects conferring in text and links them. Then the link is created between mentions

and DBpedia URIs. Output is aligned in the form of triples. Then these unmapped triples are mapped in DBpedia namespace. More than 1,000,000 triples were extracted from 114,000 Wikipedia articles and 189,000 triples are mapped to DBpedia namespace. After evaluation f1-measure of 0.663 is achieved. Author aims to investigate parallel corpora creation using entity linking.

In Wikipedia mostly the data is represented in the form of info boxes. Info boxes contains many attributes that represent the entity. Many of the info boxes in wikipedia are not complete and missing some template properties. In study [23] author proposed an automated system known as iPopulator which fills in the missing data of Info boxes from the article text. It extracts parts of values independently by detecting the structure of the values of attributes. Extraction of values of attributes is performed in four steps. First structure analysis is performed on values to know the structure of attribute. Then system prepares training data set from values specified in articles. Then extractors are generated by employing CFSs (Conditional Random Fields). Then extractors find the missing values attributes from articles. Test was conducted on all available info boxes templates. The average precision of 0.91, recall of 0.66 and f-measure of 0.73 was achieved by system on 1727 distinct attributes of info boxes templates.

Knowledge graphs play important role in different application like question answering, search engines, data integration, common sense reasoning and machine learning. But it suffers from quality issues and have a negative impact on its usefulness because of erroneous assertion and constraint violation. The study [24] proposed a method to correct assertion of objects with erroneous entities/literals. The presented framework is independent of external information or KG meta data. It uses consistency reasoning and deep learning to correct informally annotated literals and erroneous objects. This method corrects the object by using related entity. Assertion of two KGs which are KG from medical domain (MED-Ent) and DBpedia with cross domain knowledge (DBP-Lit) is evaluated. This method achieved the Recall of 0.882 for DBP-Lit and 0.797 for MED-Ent.

Mostly the data in the world exist in the form of tables such as spreadsheets,

HTML tables or datasets. A lot of research is done to clean, recognize and capture these tables. Searching the tables is suitable as tables have relational nature. In tables relationship of entities is described by rows. It helps to extend the Knowledge graphs. Extending the knowledge graphs is known as KG Completion. The study [25] proposed the method of KG completion through tables. The main problem here is table interpretation. Many studies focused on the information already available in knowledge graphs. In this research two operations are performed for interpretation. First is to link row with knowledge graph entity or link table with a class and second is to associate column with knowledge graph relation. After interpretation triples are extracted from tables and added to knowledge graph. After that slot-fitting is performed in which new facts are filled in empty blocks.

Knowledge graphs are used for several tasks in data mining, machine learning, named entity linking etc. It is the network of heterogeneous information. It is used for many practical applications/tasks but its completeness and correctness are not guaranteed. The study [26] addressed completeness problem as ranking problem. This study proposed neural network that is called ProjE. This network project entities into vectors. Ranking score of triples is calculated and ordered in descending order. This approach has two layers and only connects directly connected paths of length one. It doesn't have much parameters and pre-trained embeddings are not required. This approach shows 37% improvement on facts checking tasks. In future complicated models like CNN or RNN can be implemented with presented model.

Knowledge graphs are used in many tasks like natural language processing and information retrieval. But these must be complete to utilize them more efficiently. The study [27] provides the survey of studies which uses link prediction task to complete the knowledge graphs. Semantic Information based, Translation Distance Base Model, and Neural network-based models are discussed in this survey. After experiments author concluded that neural network with less parameter and good structure perform well. Accuracy can be improved if multi-hop domain information can be aggregated. Use of additional information like node types, node attributes, prior knowledge, relationship types can improve the performance of

model. This survey used two datasets FB15k-237 and FB15K.

Table 2.2 shows the contribution, strength and weakness of existing studies related to knowledge graph completion.

TABLE 2.2: Analysis of studies focusing on knowledge graph completion

Contribution	**Strength**	**Weakness**
[20] Provides the tool named ORE to repair and enrich the ontologies. This tool detects the problem in knowledge graphs and provide guidance to repair it.	They provided a systematic approach for detecting ontology related errors.	Errors in instances could not be discovered by this tool.
[23] Proposed an automated system known as iPopulator which fills in the missing data of Info boxes from the article text.	They used the data from same entity page and didn't rely on filling data based on closeness with other entities.	Missing DBpedia classes for wikipedia info boxes are not added.
[24] Proposed a method to correct assertion of objects having erroneous entities/literals by using related entities.	They used related entities rather than considering all entities of same class.	They did not focus on enriching the DBpedia with new classes which are present in DBpedia.
[25] Proposed the method of KG completion through tables by interpreting tables and slot-fitting.	Tables are the most authentic way of mentioning concrete information in any article other than info box, they used tables to fill the missing data.	They didnt discover the incomplete mappings and classes.

Contribution	Strength	Weakness
[26] Addressed incompleteness problem as ranking problem. This study proposed neural network that is called ProjE which project entities into vectors.	One of very few studies which used neural network model to discover incompleteness problem automatically.	The proposed approach was focused on discovering incompleteness problem in instances only.

2.3 Research Gap

The considerable research has been done to remove errors, however the studies were not successful in discovering the hierarchical errors, missing concepts and relations (which were present in Wikipedia) in DBpedia ontology. To evaluate whether errors still exist or not, semantically enriched complex queries were designed and executed. It was found that these errors still exist in DBpedia.

1. There is a need to discover the source of inconsistency and incompleteness in DBpedia core components.

2. Ontology needs to be enriched with missing concepts and relations as per domain knowledge

3. Domain/Range of different properties should be modified to remove errors.

4. More specifically following questions need to be answered:

   (a) What are the reasons of inconsistency and incompleteness in DBpedia?

   (b) How to remove these errors in order to have consistent and complete KG?

   (c) What are the missing concepts and relations in DBpedia ontology and how to enrich it?

Summary

This chapter discussed the literature review and identified the research gap. Literature was categorized into two sections. First section was related to studies which

focused on discovering and removing the inconsistencies and second section was related to studies which focused on knowledge graph completion.

Chapter 3

Discovery, Analysis and Correction of Errors

3.1 Introduction

DBpedia knowledge graph needs to be investigated to find inconsistency and incompleteness. There are two ways to investigate a KG, one is by executing different queries to get the false results in instances and second is by anaylzing the DBpedia ontology to get the possible errors in ontology. Both of these ways are used in this study to discover the errors in DBpedia knowledge graph. Inconsistency and Incompleteness are very difficult to discover in Knowledge graph because they require real world knowledge therefore a lot of automatic approaches are not feasible to discover those errors. The proposed approach comprise of four steps, discovery of errors, analysis of errors, solution of errors and evaluation of errors. Finally, the solution proposed is evaluated by using the semantically enriched complex queries which generated the errors initially. If the queries are successful in fetching the desired results, it will prove that the proposed solution is correct. Errors were discovered by following two methods:

1. Creation of semantically enriched complex queries which fetch inconsistent and/or incomplete instances.
2. Analysis of Ontology to discover inconsistent and/or incomplete concepts and relations.

Figure 3.1 shows the structure of experiments performed in discovering errors.

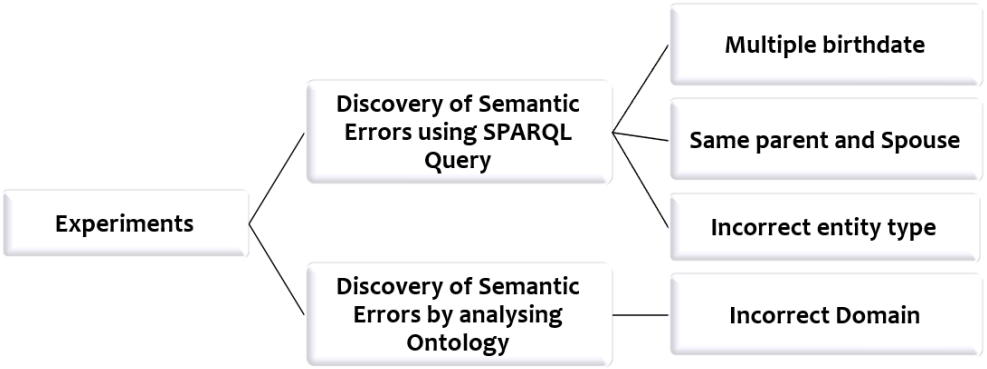

FIGURE 3.1: Discovery phase experiments

The experiments will be performed to answers following research questions.

1. What are the reasons of inconsistency and incompleteness in DBpedia?
2. How to remove these errors in order to have consistent and complete semantic web?
3. What are the missing concepts and relations in DBpedia ontology and how to enrich it?

We will answer these research questions by first discovering the semantic errors using semantically enriched complex queries and analysis of ontology then we will find the reasons for these errors and implement solutions to correct them. The structure of the experiments is such that discovery, analysis and solution are grouped according to the nature of issues/problems. In section 3.2 semantic errors are discovered using semantically enriched complex queries. In section 3.3 semantic errors are discovered by analyzing the ontology.

Figure 3.2 shows the experimental flow of each experiment.

FIGURE 3.2: Experimental flow

3.2 Discovery of Semantic Errors Using Semantically Enriched Complex Logic

DBpedia is an open access knowledge graph and provides a public endpoint for queries. One way to discover semantic errors is by designing semantically enriched complex queries which fetch contradictory results. The intuition behind this approach was to fetch results from DBpedia which could not be possible in real life. By designing these semantically enriched complex queries, different inconsistent and incomplete instances were discovered.

3.2.1 Multiple Birth Dates Logic for Discovering Inconsistency

In real life every living thing have one birth date, so if any entity in DBpedia have more than one birth date then the results are not consistent. Therefore below mentioned semantically enriched complex query was designed to fetch the list of those entities which have more than one birth date along with number of birth dates. The number of inconsistent records which were retrieved by this query was 417.

```
SELECT ?person count(?dates)
        WHERE {
                ?person dbo:birthDate ?dates.
        }
Group by ?person
HAVING (COUNT(?dates) > 1)
```

3.2.1.1 Discovery of Multiple Birth Dates Inconsistency

BirthDate must be unique for each entity, however it was discovered that some of DBpedia entities are showing multiple such properties. Table 1.2 shows the DBpedia profile of entity Karen McCarron. It shows the inconsistency where Karen have more than one birth dates which are 1968-12-20 and 2002-07-22. If

any query or operation by any user will be performed using birth date property, then the results might be inconsistent because many of the entities will return more than one birth dates. Such inconsistencies must be removed to make the DBpedia a true semantic web.

3.2.1.2 Analysis of Multiple Birth Dates Inconsistency

To know where the error exists Wikipedia page of Karen was checked where it was found that Karen McCarron page contains two info boxes, one for Karen McCarron and other for her child Kate McCarron. Birth dates of both parent and child are assigned to Karen McCarron in DBpedia. It was observed that all those Wikipedia entities which have more than one info box are being merged into one resource by DBpedia framework. In this way the location of error was traced which is the conversion process and in particular DBpedia framework. Table 3.1 and table 3.2 shows the Wikipedia info boxes of Karen McCarron and Kate McCarron. Karen McCarron's birth date is 1968-12-20 and Kate McCarron's birth date is 2002-07-22 but on DBpedia both of these birth dates are associated to Karen McCarron.

TABLE 3.1: Wikipedia info box Karen Frank-McCarron [9]

Info box title	**Karen Frank McCarron**
Born	Karen Frank December 20, 1968 (age 53) West Germany
Other names	Karen McCarron
Alma mater	University of Illinois at Urbana–Champaign Southern Illinois University School of Medicine0
Occupation	Pathologist
Criminal status	Incarcerated
Spouse(s)	Paul McCarron (c. 1995–2006, div.)

TABLE 3.2: Wikipedia info box Kate McCarron [9]

Info box title	**Kate McCarron**
Born	Katherine Marie McCarron July 22, 2002 Peoria, Illinois, U.S.
Died	May 13, 2006 (aged 3) Morton, Illinois, U.S.
Cause of death	Suffocation
Resting place	Resurrection Cemetery and Mausoleum, Peoria, Illinois, U.S.
Nationality	American German
Known for	Murder victim
Parent(s)	Karen Frank-McCarron Paul McCarron

3.2.1.3 Discovery of Self Parents Inconsistency

By looking at the results of multiple birth dates query different inconsistencies were discovered. One inconsistency was that person was also mentioned as parent of himself/herself. It can be seen that Person Karen McCarron is mentioned as the parent of herself in the Table 3.3. When end user query any information related to parent property, the results might be contradictory as different entities mentioned their own self as parent in DBpedia. In table 3.3 it can be seen in the DBpedia profile of Karen McCarron that parents of Karen McCarron are Paul McCarron and Karen Frank-McCarron. In real life a person can not be a parent of himself/herself and also when someone query about parents of parents then this

will create a loop where entity will point to itself as parents of parents, therefore this inconsistency must be removed.

TABLE 3.3: Self parent inconsistency [8]

DBpedia page	**Karen McCarron**
Occupation	Pathologist
Other names	Karen McCarron
Parents	Paul McCarron Karen Frank-McCarron
Resting place	Resurrection Cemetery and Mausoleum, Peoria, Illinois, U.S.

3.2.1.4 Analysis of Self Parents Inconsistency

To understand the reason that why Karen McCarron is mentioned as parent of herself, her Wikipedia profile was checked and it was found that as Kate McCarron is child of Karen and info box of Kate have parent Karen therefore when these two info boxes are combined in one resource in DBpedia then the parent of child also becomes the parent so in this way an entity Karen McCarron becomes parent of herself. The reason behind this inconsistency is also same as multiple birth dates inconsistency which is merging of multiple info boxes into single resource in DBpedia.

3.2.1.5 Discovery of Same Values for Distinct Properties Inconsistency

Another inconsistency discovered using multiple birth dates query was that the properties which must have distinct values in real life were having same values. Table 3.4 shows the example where spouse and parent of enitity Karen McCarron is same which is Paul McCarron.

TABLE 3.4: Same values for distinct properties inconsistency [8]

DBpedia page	**Karen McCarron**
Spouse	Paul McCarron
Other names	Karen McCarron
Parents	Paul McCarron Karen Frank-McCarron
Resting place	Resurrection Cemetery and Mausoleum, Peoria, Illinois, U.S.

3.2.1.6 Analysis of Same Values for Distinct Properties Inconsistency

As the wikipedia page of entity Karen McCarron contains two info boxes one for parent Karen and other for her child Kate, the parent property of Kate and the spouse property of Karen have the same value because father of Kate is also the husband of Karen. Therefore when these info boxes were merged into single resource in DBpedia, all these properties were associated to Karen.

3.2.1.7 Discovery of Incorrect Association of Information Inconsistency

To discover further errors, DBpedia page and Wikipedia info box of inconsistent entities were compared and it was found that on DBpedia some entities have such properties which are not listed in their Wikipedia info box, infact these properties belong to some other info box in Wikipedia. One example of this error is Helene Demuth. Table 3.5 and table 3.6 shows the DBpedia page and Wikipedia info box of entity Helene Demuth. The Wikipedia info box of Helene Demuth doesn't mention birthName but on DBpedia page birthName property is present which is in fact the birthName of another entity Frederick Lewis Demuth.

TABLE 3.5: Wikipedia info box Helene Demuth [9]

Info box title	**Helene Demuth**
Born	31 December 1820 Sankt Wendel, Principality of Lichtenberg, Duchy of Saxe-Coburg-Saalfeld, German Confederation
Died	4 November 1890 (aged 69) London, United Kingdom
Resting place	Highgate Cemetery
Nationality	Prussian, German
Known for	Housekeeper of Karl Marx, later household manager and political confidante of Frederich Engels

TABLE 3.6: DBpedia page Helene Demuth [8]

DBpedia page	**Helene Demuth**
Abstract	Helene Demuth (31 December 1820 – 4 November 1890) was a German housekeeper who worked for Jenny and Karl Marx, and later served as the household manager and political confidante of Friedrich Engels.
Birth date	1820-12-31 1851-06-23
Birth name	Henry Frederick Demuth

3.2.1.8 Analysis of Incorrect Association of Information Inconsistency

This inconsistency was discovered by comparing the Wikipedia info box of Helene Demuth with her DBpedia page. On further analysis of Wikipedia page it was found that there also exists more than one info boxes, Helene Demuth and her child Frederick Lewis Demuth. Helene Demuth doesn't mention birthName but when both info boxes are merged, birthName of Frederick is assigned to Helene in DBpedia. So this inconsistency also arises due to merging of multiple info boxes into a single resource.

3.2.1.9 Solution for Inconsistencies from Multiple Birth Dates Logic

Following four types of inconsistencies were discovered from multiple birth dates query:

1. Multiple birth dates
2. Self parents
3. Same values for distinct properties
4. Incorrect association of information

It was found that the reason for all these inconsistencies is same which is merging of multiple Wikipedia info boxes into single resource in DBpedia. The component of DBpedia which is responsible for this error is DBpedia framework. To remove this error the solution is proposed that if the info boxes have different names with respect to title of page then these must be stored as separate resource in DBpedia. The process for modification of extraction framework is as follows:

1. Clone https://github.com/dbpedia/extraction-framework/ github repository.
2. Create new branch from Dev branch.
3. Make changes in the Extraction Framework.

4. Commit Changes

5. Send merge request after completion.

To implement this solution following steps have been performed:

1. Installation of DBpedia extraction framework

2. Understanding of different DBpedia extractors code.

3. Modifications of TemplateMappings.scala file to implement our solution.

   (a) Checking if name of info box is different

```
//checking if node is an infobox and have different name
val isInfobox = if (node.title.decoded.contains("Infobox")) {
  true
} else {
  false
}

var condition4_same_entity_infoBox = true

if(isInfobox)
{
  //getting name property from infobox
  val allNames = node.children.filter(p => p.key == "name")
  var name = subjectUri;
  if(allNames.size > 0)
    name = allNames(0).propertyNodeValueToPlainText

  //getting subject  of wikipedia page
  var splittedURI = subjectUri.split("/")
  var pageTitle = splittedURI(splittedURI.size - 1)

  if(!name.contains(pageTitle) && !pageTitle.contains(name))
    condition4_same_entity_infoBox = false
}
```

(b) Saving it as separate resource if names are different

```
// If all above conditions are met then use the main
resource, otherwise create a new one
val instanceUri = {
  if ( (!condition1_create_correspondingproperty) &&
  (!condition2_template_exists) && condition3_subclass &&
  condition4_same_entity_infoBox) subjectUri
  else generateUri(subjectUri, node)
}
```

3.2.1.10 Evaluation of Proposed Solution

After making the proposed changes to remove inconsistencies from multiple birth dates query, it must be verified if the inconsistencies were removed. For this DBpedia extractor was ran in adhoc mode for single entity extraction and Karen McCarron page was extracted using the updated code as seen in Figure 3.3

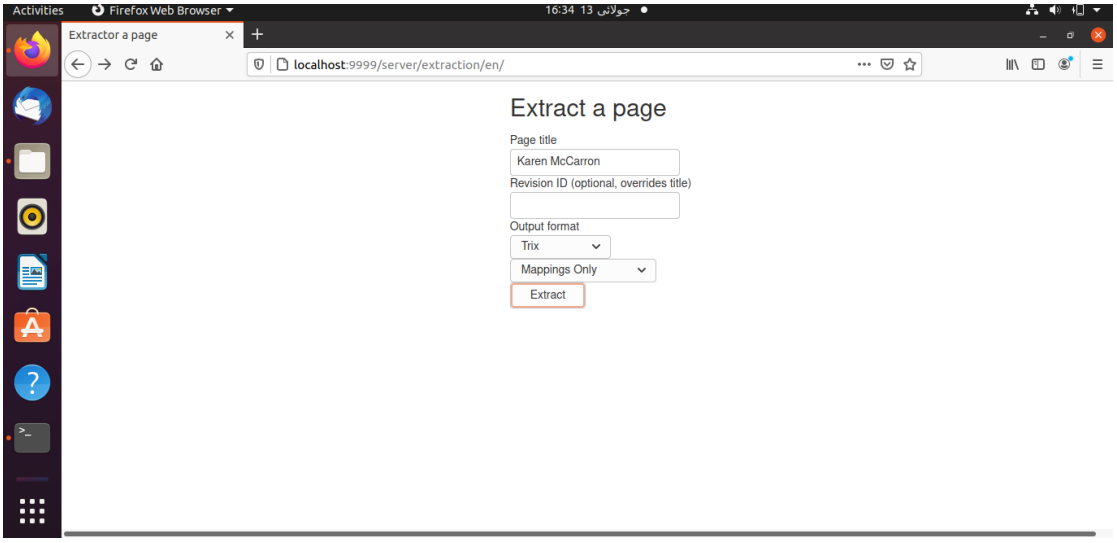

FIGURE 3.3: DBpedia framework adhoc page for Karen extraction

The results of this extraction process are shown in table 3.7. After extraction it can be seen that the Karen McCarron and Kate McCarron are now stored as

separate resources and all four inconsistencies from multiple birth dates query are hence removed.

TABLE 3.7: DBpedia framework Karen extraction result

Resource	Property	Value
http://dbpedia.org/resource/Karen_McCarron	http://dbpedia.org/ontology/birthDate	1968-12-20
http://dbpedia.org/resource/Karen_McCarron	http://dbpedia.org/ontology/birthPlace	http://dbpedia.org/resource/West_Germany
http://dbpedia.org/resource/Karen_McCarron__Kate_McCarron__1	https://xmlns.com/foaf/01/name	Kate McCarron
http://dbpedia.org/resource/Karen_McCarron__Kate_McCarron__1	http://dbpedia.org/ontology/birthDate	2002-07-22
http://dbpedia.org/resource/Karen_McCarron__Kate_McCarron__1	http://dbpedia.org/ontology/birthYear	2002

3.2.2 Same Parent and Spouse Logic for Discovering Inconsistency

The below mentioned query is designed to fetch those Person entities who have same parent and spouse. It returns the list of Person entities having same resource as Parent and Spouse stored in knowledge graph. In real life parent and

spouse cannot be same entity, therefore the above query was designed to fetch the inconsistent results. The total number of inconsistent entities retrieved are 44.

```
SELECT ?person, ?parent
        WHERE {
                ?person a dbo:Person.
                ?person dbo:parent ?parent.
                ?person dbo:spouse ?parent.
        }
```

3.2.2.1 Discovery of Same Parent and Spouse Inconsistency

Spouse and parent are two distinct relations and one cannot be parent and spouse of the same person, therefore a query was designed which fetched those entities from DBpedia whose parent and spouse is same. Table 3.8 shows the list of few person entities in DBpedia who have same parent and spouse.

TABLE 3.8: Same parent and spouse query results

Person	Parent and Spouse
http://dbpedia.org/resource/Ugyen_Wangchuk	http://dbpedia.org/resource/Ashi_(title)
http://dbpedia.org/resource/Seru_Epensia_Cakobau	http://dbpedia.org/resource/Adi_(title)
http://dbpedia.org/resource/Nebettawy	http://dbpedia.org/resource/Ramesses_II
http://dbpedia.org/resource/Glayton_Modise	http://dbpedia.org/resource/Frederick_Samuel_Modise

Person	Parent and Spouse
http://dbpedia.org/resource/Gottfried_Graf_von_Bismarck-Schonhausen	http://dbpedia.org/resource/House_of_Hoyos
http://dbpedia.org/resource/Khalid_bin_Hamad_Al_Khalifa	http://dbpedia.org/resource/Sheikh
http://dbpedia.org/resource/Rafi-ush-Shan	http://dbpedia.org/resource/Nur-un-nissa_Begum
http://dbpedia.org/resource/Amun-her-khepeshef	http://dbpedia.org/resource/Nefertari
http://dbpedia.org/resource/Mikhail_Prince_of_Abkhazia	http://dbpedia.org/resource/House_of_Dadiani
http://dbpedia.org/resource/Lord_Arthur_John_Henry_Somerset	http://dbpedia.org/resource/Elizabeth_Boscawen
http://dbpedia.org/resource/Shery_(Egypt)	http://dbpedia.org/resource/Inet_(woman)

3.2.2.2 Analysis of Same Parent and Spouse Inconsistency

Instances of query result were analyzed that when names are written with title prefix in Wikipedia then the DBpedia framework only stores first part which is the title and skips the actual name of person. So there must be something wrong in the DBpedia framework. Table 3.9 shows the Wikipedia profile of entity Jigme Dorji Wangchuck which have spouse Ashi Kesang Choden and parents Jigme Wangchuck, Ashi Phuntsho Choden where Ashi is the title in Bhutanese language for mature women [9].

TABLE 3.9: Wikipedia info box Jigme Dorji Wangchuck [9]

Info box title	**Jigme Dorji Wangchuck**
Reign	30 March 1952 – 21 July 1972
Coronation	27 October 1952
Predecessor	Jigme Wangchuck
Successor	Jigme Singye Wangchuck
Born	2 May 1929 Thruepang Palace, Trongsa
Died	21 July 1972 (aged 43)
Burial	Cremated at Kurjey Lhakhang
Spouse	Ashi Kesang Choden
Issue	Sonam Choden Wangchuk Dechen Wangmo Wangchuk Jigme Singye Wangchuck Pema Lhaden Wangchuk Kesang Wangmo Wangchuk
House	Wangchuck
Father	Jigme Wangchuck
Mother	Ashi Phuntsho Choden
Religion	Buddhism

Where as in the DBpedia profile for Jigme Dorji Wangchuck spouse and parent is Ashi(title) so it was concluded that DBpedia framework is skipping the actual name and storing only title of entities which have some reference page in the Wikipedia. Table 3.10 shows the DBpedia profile of Jigme Dorji Wangchuck where it can be seen that parent and spouse have same value which is Ashi_(title).

TABLE 3.10: DBpedia page Jigme Dorji Wangchuck [8]

DBpedia page	**Jigme Dorji Wangchuck**
Parent	Jigme Wangchuk Ashi_(title)
Predecessor	Jigme Wangchuk
Spouse	Ashi_(title)
Successor	Jigme Singye Wangchuck

On further analysis of DBpedia extraction framework and Wikipedia info boxes it was found that there are two reasons behind the errors discovered by same parent and spouse query which are discussed below:

1. Extraction Rule:

By checking the backend of Wikipedia pages of different entities it was observed that on some Wikipedia pages single quotation marks are used as Separators in any property value. On the basis of these separators, property value must split. Table 3.11 shows the backend of Wikipedia page of Jigme Dorji Wangchuck where two single quotation marks are used as value separators in spouse and mother property.

TABLE 3.11: Wikipedia info box backend Jigme Dorji Wangchuck [9]

Infobox	**Monarch**
Name	Jigme Dorji Wangchuck [[File:Jigme Dorji Wangchuck Name.svg]]
Title	[[List of rulers of Bhutan|King of Bhutan]]
Succession	[[Druk Gyalpo|King of Bhutan]]
Reign	30 March 1952 – 21 July 1972
Predecessor	[[Jigme Wangchuck]]
Successor	[[Jigme Singye Wangchuck]]
Spouse	' '[[Ashi (title)|Ashi]]' ' [[Kesang Choden (born 1930)|Kesang Choden]]
Issue	[[Sonam Choden Wangchuck]] [[Dechen Wangmo Wangchuck]] [[Jigme Singye Wangchuck]] [[Pema Lhaden Wangchuck]] [[Kesang Wangmo Wangchuck]]
House	[[House of Wangchuck|Wangchuck]]
Father	[[Jigme Wangchuck]]
Mother	' '[[Ashi (title)|Ashi]]' ' [[Phuntsho Choden]]
Religion	[[Buddhism]]

Although two single quotation marks are used as Separators in Wikipedia but the

extraction rule used in extraction framework for splitting value doesn't contain single quotation marks. Following lines of code shows the extraction rule used for splitting property value. Therefore DBpedia only take Ashi as property value and skips the actual name part. The class responsible for splitting value is DataParserConfig.scala.

```
val splitPropertyNodeRegexObject   = Map (
    "en" -> """<br\s*\/?>|\n| and | or | in |/|;|,"""
  )
```

2. Post Processing Issue::

It was found that DBpedia contains property values which are violating range of property. For example range of spouse is Person but still DBpedia contains spouse relations with entities which are not Person. This indicates some issue in framework where domain and range of statement is checked. DBpedia framework have a separate module of post processing the statements to filter out any statement which is violating domain and range and also check for disjoined axioms. The class responsible for this purpose is TypeConsistencyCheck.scala. In this class an issue was found that currently all the statements are passed to regular set even if they are violating domain and range and only disjoint statements are filtered out and stored in datasets for incorrect statements. Following is the code snippet from TypeConsistencyCheck.scala file where datasets are assigned to types of statements. It can be seen that untyped and nondisjoint statements are also stored in correct dataset.

```
val correctDataset: Dataset = DBpediaDatasets.OntologyPropertiesObjectsCleaned
val disjointRangeDataset: Dataset  =
DBpediaDatasets.OntologyPropertiesDisjointRange
val untypedRangeDataset: Dataset = correctDataset
val nonDisjointRangeDataset: Dataset = correctDataset
val disjointDomainDataset: Dataset  =
DBpediaDatasets.OntologyPropertiesDisjointDomain
val untypedDomainDataset: Dataset = correctDataset
val nonDisjointDomainDataset: Dataset = correctDataset
```

3.2.2.3 Solution for Same Parent and Spouse Inconsistency

There are two main reasons for this inconsistency which are discussed above, to remove this inconsistency first extraction rule and then post processing issue was resolved.

1. Extraction Rule Correction:

To include the two single quotation marks as separators extraction rule was changed to:

```
val splitPropertyNodeRegexObject   = Map (
  "en" -> """<br\s*\/?>|\n| and | or | in |/|;|''|,"""
)
```

The new extraction rule also contains two single quotation marks for splitting value .

2. Post Processing Correction:

Previously non-disjoint and untyped statements were also stored to correct dataset which was causing problems, to eliminate this incorrect triples were passed to other incorrect datasets. Following is the updated code in which untyped statements are stored in untypedDataset and nondisjoint statements which are violating domain and range are stored in nondisjointDataset

```
   val correctDataset: Dataset = DBpediaDatasets.OntologyPropertiesObjectsCleaned
val untypedDataset:Dataset = new Dataset("mappingbased- properties-untyped")
val nondisjointDataset:Dataset = new Dataset("mappingbased-properties-non-disjoint")
 val disjointRangeDataset: Dataset  = DBpediaDatasets.OntologyPropertiesDisjointRange
 val untypedRangeDataset: Dataset =  untypedDataset
val nonDisjointRangeDataset: Dataset = nondisjointDataset
 val disjointDomainDataset: Dataset  =
 DBpediaDatasets.OntologyPropertiesDisjointDomain
 val untypedDomainDataset: Dataset = untypedDataset
 val nonDisjointDomainDataset: Dataset = nondisjointDataset
```

3.2.2.4 Evaluation of Proposed Solution

After impementing the proposed changes, same parent and spouse query was executed again and the result set of inconsistent records is reduced from 44 to 4, 3 of which are due to wrong classification of entities by extractors and reason for one of them is unknown to us. Table 3.12 shows the list of inconsistencies which were not removed after implementing proposed changes.

TABLE 3.12: Remaining inconsistent records of same parent and spouse query

Person	Parent and Spouse
http://dbpedia.org/resource/Gottfried_Graf_von_Bismarck-Schonhausen	http://dbpedia.org/resource/House_of_Hoyos
http://dbpedia.org/resource/Benjamin_Bates_IV	http://dbpedia.org/resource/Bates_family
http://dbpedia.org/resource/Aung_Lwin	http://dbpedia.org/resource/Burmese_name
http://dbpedia.org/resource/Frank_John_William_Goldsmith	http://dbpedia.org/resource/Nee

3.2.3 Incorrect Entity Type Logic for Discovering Incompleteness

Another semantically enriched complex query is designed to fetch those entities who are not person but still contain properties which are related to person. The

query returns an entity if it is not person but still have spouse property. Further different filters are used in this query to filter out false results.

```
SELECT Distinct count(?person)
WHERE {
        ?person dbp:spouse ?val.
        ?person a ?type.
        FILTER NOT EXISTS {?person a dbo:Person}.
        FILTER NOT EXISTS {?person a dbo:FictionalCharacter}.
        FILTER (!strstarts(str(?type), str"(http://dbpedia.org/class/yago))).
}
```

3.2.3.1 Discovery of Incompleteness

It was found that there are a lot of entities in DBpedia which contain family relation properties but are classified as owl:Thing instead of dbo:Person. Due to this many inaccurate results are fetched when queried because family relations like spouse should only be applicable to dbo:Person and not owl:Thing. Thousands of such entities are wrongly classified. Above query lists those entities which have spouse relation but are classified as owl:Thing. The incorrect entity type query returned 4885 results, few of them are shown in the table 3.13.

TABLE 3.13: Incorrect entity type query results

Person
https://dbpedia.org/resource/Barbara_Erickson_London
https://dbpedia.org/resource/Barbara_Fallis
https://dbpedia.org/resource/Baruch_Mordechai_Ezrachi
https://dbpedia.org/resource/Beatrice_Beebe
https://dbpedia.org/resource/Barbara_Schneider_(sociologist)
https://dbpedia.org/resource/Barbara_Scholz
https://dbpedia.org/resource/Baruch_Mordechai_Ezrachi
https://dbpedia.org/resource/Beatrix_Tugendhut_Gardner
https://dbpedia.org/resource/Ben_Ainslie

Person
https://dbpedia.org/resource/Benard_E._Aigbokhan
https://dbpedia.org/resource/Benjamin_Church_(physician)
https://dbpedia.org/resource/Bunyan_Bryant
https://dbpedia.org/resource/Buster_Welch
https://dbpedia.org/resource/C._Alfred_%22Chief%22_Anderson
https://dbpedia.org/resource/C._George_Boeree
https://dbpedia.org/resource/C._J._Ryan
https://dbpedia.org/resource/Capitan_Vicente_Almandos_Almonacid

Table 3.14 shows the DBpedia profile of entity Lucie Fink where it can be seen that Lucie Fink is classified as owl:Thing and have properties which are related to Person like spouse, nationality etc.

TABLE 3.14: DBpedia profile of Lucie Fink [8]

DBpedia page	**Lucie Fink**
Type	owl:Thing
Abstract	Lucinda Beatrice Fink (born August 3, 1992), better known as Lucie Fink, is an American YouTube personality and lifestyle host. As of April 2021, her channel has approximately 22.7 million video views and 280,000 subscribers.
Spouse	2019-09-21 Michael J. Morris
Website	https://luciefink.com/

3.2.3.2 Analysis of Incompleteness

These incompleteness errors are about getting list of entities who are not person but contains Person properties like spouse. In the Wikipedia profile of such entities, it was observed that these entities belong to info boxes against which there are no DBpedia classes and hence are not mapped to DBpedia. Table 3.15 shows the Wikipedia profile of Lucie Fink (YouTube Personality). On Wikipedia this entity is YouTube personality while on DBpedia entity is classified as owl:Thing because there exist no class for YouTuber in DBpedia.

TABLE 3.15: Wikipedia page backend of Lucie Fink [9]

Infobox	**YouTube personality**
Name	Lucie Fink
Birth name	Lucinda Beatrice Fink
Birth date	birth date and age|1992|8|3
Birth place	[[White Plains, New York]], U.S.
Nationality	[[United States|American]]
Spouse	marriage|Michael J. Morris|September 21, 2019
Website	URL|https://luciefink.com/

Beside Info box YouTube personality, other info boxes were also discovered by analyzing the results against which either there exist no DBpedia class or mappings are missing. Table 3.16 shows those info boxes against which there exist no DBpedia class and no mappings as well.

Table 3.16: Wikipedia info boxes with no classes in DBpedia

Wikipedia Info boxes with no classes in DBpedia
Info box YouTube Personality
Info box Academic
Info box Sailor
Info box Jewish leader
Info box aviator
Info box Spy
Info box rebbe
Info box pretender
Info box Latter Day Saint Biography
Info box Pharaoh
Info box Native American Leader
Info box Police Officer

However for following info boxes, there exist classes in DBpedia, only mappings are missing:

Table 3.17: Wikipedia info boxes with no mappings in DBpedia

Wikipedia Info boxes with no mappings in DBpedia
Info box Dancer
Info box Deity
Info box person/Wikidata

3.2.3.3 Solution for Incompleteness

After the analysis phase it was known that DBpedia doesn't contain some classes which are responsible for these types of errors. To overcome this problem, new DBpedia classes need to be introduced in DBpedia and after that the unmapped Wikipedia info boxes should be mapped to newly created DBpedia classes . By

following the mentioned approach, Youtuber class was introduced in DBpedia and then info box Youtube Personality was mapped to Youtuber class. The process for creation of Ontology classes and properties is as follows:

1. Creation of account on http://mappings.dbpedia.org/index.php/Special:UserLogin2
2. Another account on http://forum.dbpedia.org
3. Request for editorial rights on https://forum.dbpedia.org using username.
4. Creation of new class template and new property template on http://mappings.dbpedia.org/index.php?title=Special%3AAllPages&from=&to=&namespace=200

The details of newly created class in DBpedia is shown in table 3.18.

TABLE 3.18: Wikipedia page backend of Lucie Fink [9]

Ontology class	
rdfs:label (en)	Youtuber
rdfs:label (eu)	youtuberra
rdfs:label (da)	youtuber
rdfs:label (fr)	youtubeuse
rdfs:label (hi)	यूट्यूबर
rdfs:label (ar)	اليوتيوب
rdfs:comment (en)	a person who uploads, produces, or appears in videos on the video-sharing website YouTube
rdfs:subClassOf	Person

The process of mapping any wikipedia info box to DBpedia class is as follows:

1. Step 1 to step 3 of ontology classes creation process.
2. Retrieve encoded template page name from Wikipedia
3. Create a wiki page in DBpedia mapping wiki using encoded template.
4. Define the ontology class of DBpedia to which we want to map the template.

Table 3.19 shows the details of mappings created for info box Youtube Personality.

TABLE 3.19: Info box youtube personality mapping to youtuber class

Template mapping	
mapToClass	Youtuber
mappings	PropertyMapping | templateProperty = name | ontologyProperty = foaf:name PropertyMapping | templateProperty = website | ontologyProperty = foaf:homepage PropertyMapping | templateProperty = $birth_date$|$ontologyProperty = birthDate$ PropertyMapping | templateProperty = nationality | ontologyProperty = nationality

By following the same process, all other missing classes of discovered info boxes are created and then all the unmapped info boxes are mapped to DBpedia classes. 12 new classes are created and 15 info boxes are mapped with just few basic properties.

3.2.3.4 Evaluation of Proposed Solution

After creating the required classes and their mappings, incorrect entity type query was again executed to fetch updated results which are reduced from 4885 to 72. The reason for remaining 72 inconsistencies are related to DBpedia extractors which are wrongly classifying the instances. Table 3.20 shows some of the remaining entities which are incomplete .

TABLE 3.20: Remaining incomplete instances

Remaining incomplete instances
https://dbpedia.org/resource/Bharata_chakravartin
https://dbpedia.org/resource/Chakreshvari
https://dbpedia.org/resource/Manasa
https://dbpedia.org/resource/Nabhi
https://dbpedia.org/resource/Oshun
https://dbpedia.org/resource/Padmavati_(Jainism)
https://dbpedia.org/resource/Rishabhanatha
https://dbpedia.org/resource/Robert_William_Fisher
https://dbpedia.org/resource/Shango
https://dbpedia.org/resource/Shiva
https://dbpedia.org/resource/Taylor_Winterstein

3.3 Discovery of Semantic Errors by Analysis of Ontology

One way to discover semantic errors is by analysis of ontology. In this process properties definition, their domain range and their class hierarchy was observed for any inconsistency.

3.3.1 Discovery of Incorrect Domain Inconsistency

It was discovered that domain of birthDate property is restricted to Person class which is not true in real life because every living thing have a birthDate. Table 3.21 shows the detail of birthDate property in DBpedia where it can be seen that its domain is Person.

TABLE 3.21: Birth date property data [8]

Ontology data type property	
rdfs:label (en)	Birth date
rdfs:label (bn)	জন্মদিন
rdfs:label (ca)	data de naixement
rdfs:label (de)	Geburtsdatum
rdfs:label (el)	ημερομηνία_γέννησης
rdfs:label (fr)	date de naissance
rdfs:label (ga)	dáta breithe
rdfs:label (ja)	生年月日
rdfs:label (nl)	geboortedatum
rdfs:label (pl)	data urodzenia
rdfs:domain	Person
rdfs:range	xsd:date
rdf:type	owl:FunctionalProperty

It can be seen that domain of birthDate property is Person, similarly domain of deathDate and other similar properties like birthPlace is also Person which in real world is not correct. To further strengthen the view source data was checked which is Wikipedia info box and it was found that there also exist Animal info boxes which contain these properties like birthDate so restricting its domain to Person is somehow incorrect. Following is the list of properties of info box Animal template where birthDate and birthPlace properties also exist.

TABLE 3.22: Wikipedia info box animal

Wikipedia Info boxe Animal
name
image
Image_upright
Caption
Othername
Species
Breed
gender
birth_name
birth_date
birth_place
death_date
death_place
death_cause
resting_place

3.3.2 Analysis of Incorrect Domain Inconsistency

By the analysis of DBpedia ontology it was observed that Person class is listed under Agent class and Animal class is listed under Eukaryote class in the Species class. As these two classes contain a lot of similar properties so they must be grouped under single parent class and domain of such similar properties should be that parent class. Figure 3.4 shows the current hierarchy of these classes in DBpedia.

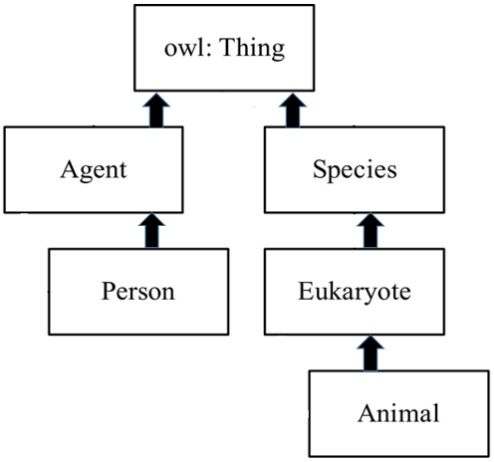

FIGURE 3.4: DBpedia class hierarchy

3.3.3 Solution of Incorrect Domain Inconsistency

The proposed solution is to group Person and Animal class under single parent and make domain of similar properties to that parent class. Human being is also a type of Animal so Person class must be moved to Animal section and domain of properties like birthDate, deathDate should be set to Animal class. Figure 3.5 shows the newly proposed hierarchy:

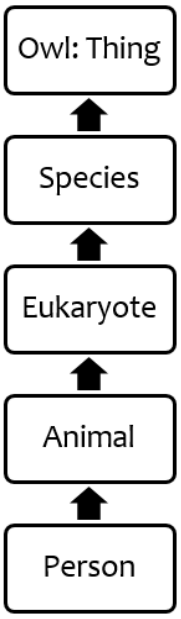

FIGURE 3.5: DBpedia updated class hierarchy

To implement this solution following steps have been performed:

1. Changing super class of Person from Agent to Animal as shown in table 3.23.

TABLE 3.23: DBpedia person class [8]

Ontology class	
rdfs:label (el)	Πληροφορίες προσώπου
rdfs:label (en)	person
rdfs:label (eu)	pertsona
rdfs:label (da)	Person
rdfs:label (ur)	شخص
rdfs:label (de)	Person
rdfs:label (sl)	Oseba
rdfs:label (it)	persona
rdfs:label (pt)	pessoa
rdfs:label (fr)	personne
rdfs:label (ga)	duine
rdfs:label (es)	persona
rdfs:label (ja)	人_(法律)
rdfs:label (nl)	persoon
rdfs:label (pl)	osoba
rdfs:label (ar)	شخص
rdfs:subClassOf	Animal
owl:equivalentClass	foaf:Person, schema:Person, wikidata:Q215627, wikidata:Q5, dul:NaturalPerson

2. Changing domain of birthDate to Animal as shown in table 3.24.

TABLE 3.24: DBpedia birth date property updated data [8]

Ontology data type property	
rdfs:label (en)	Birth date
rdfs:label (bn)	জন্মদিন
rdfs:label (ca)	data de naixement
rdfs:label (de)	Geburtsdatum
rdfs:label (el)	ημερομηνία_γέννησης
rdfs:label (fr)	date de naissance
rdfs:label (ga)	dáta breithe
rdfs:label (ja)	生年月日
rdfs:label (nl)	geboortedatum
rdfs:label (pl)	data urodzenia
rdfs:domain	Animal
rdfs:range	xsd:date
rdf:type	owl:FunctionalProperty

3. Changing domain of birthPlace to Animal as shown in table 3.25.

TABLE 3.25: DBpedia birth place property data [8]

Ontology object property	
rdfs:label (en)	birth place
rdfs:label (ca)	lloc de naixement
rdfs:label (de)	Geburtsort
rdfs:label (el)	τόπος
rdfs:label (fr)	lieu de naissance
rdfs:label (ga)	áit bhreithe
rdfs:label (ja)	出生地
rdfs:label (nl)	geboorteplaats
rdfs:label (pl)	miejsce urodzenia
rdfs:comment (en)	where the person was born
rdfs:domain	Animal
rdfs:range	Place
rdfs:subPropertyOf	dul:hasLocation
owl:equivalentProperty	schema:birthPlace, wikidata:P19

4. Changing domain of deathDate to Animal as shown in table 3.26.

TABLE 3.26: DBpedia death date property data [8]

Ontology object property	
rdfs:label (en)	death date
rdfs:label (de)	Sterbedatum
rdfs:label (fr)	date de décès
rdfs:label (el)	ημερομηνία
rdfs:label (ja)	没年月日
rdfs:label (nl)	sterfdatum
rdfs:domain	Animal
rdfs:range	xsd:date
rdfs:type	owl:FunctionalProperty
owl:equivalentProperty	schema:deathDate, wikidata:P570, gnd:dateOfDeath

5. Changing domain of deathPlace to Animal as shown in table 3.27.

TABLE 3.27: DBpedia death place property data [8]

Ontology object property	
rdfs:label (en)	death place
rdfs:label (de)	Sterbeort
rdfs:label (fr)	lieu de décès
rdfs:label (el)	τόπος_θανάτου
rdfs:label (ja)	死没地
rdfs:label (nl)	plaats van overlijden
rdfs:domain	Animal
rdfs:range	Place
rdfs:comment (en)	The place where the person died
owl:equivalentProperty	schema:deathPlace, wikidata:P20
rdfs:subPropertyOf	dul:hasLocation

Summary

This chapter explained different experiments conducted to answer research questions. First semantic errors were discovered using semantically enriched complex queries which included multiple birth dates query for discovering inconsistency error, same parent and spouse query for discovering inconsistency and incorrect entity type query for discovering incompleteness. Secondly semantic errors were discovered by analysis of ontology which included domain inconsistency. Analysis of these errors had been done and solutions were provided. Figure 3.6 shows the summary of erros discovered in this study.

Errors	Total Errors	Reason for Errors
Multiple birthdates	104	Multiple info boxes
Self parents	95	Multiple info boxes
Same values for distinct properties	130	Multiple info boxes
Incorrect association of information	88	Multiple info boxes
Same parent and spouse	44	Regex and post processing issue
Incompleteness problem	4885	Missing concepts in DBpedia
Incorrect domain	4	Ontology incorrect

FIGURE 3.6: Discovered errors summary

Chapter 4

Evaluation of Results

4.1 Evaluation

DBpedia was introduced in 2007, however even after fifteen years it still contains semantic errors which were not resolved previously, this study discovered such errors and resolved them. This chapter evaluates all the corrective actions taken to remove inconsistency and incompleteness related to ontology and instances.

For evaluation of results different test cases have been designed which compares the errors before and after implementing proposed solutions. Each test case has been designed for single type of error. Same process was used for evaluation purpose which was used to discover errors. Table 4.1 illustrates description of different test cases which shows different types of inconsistencies resolved.

Table 4.1: Description of test cases

Test Case	**Description**
Test Case 1	Multiple birth dates inconsistency
Test Case 2	Self parents inconsistency
Test Case 3	Same values for distinct properties inconsistency
Test Case 4	Incorrect association of information inconsistency
Test Case 5	Same parent and spouse inconsistency
Test Case 6	Incompleteness error
Test Case 7	Incorrect domain inconsistency

4.1.1 Test Case 1 (Multiple Birth Dates Inconsistency)

This test case deals with the multiple birth dates inconsistency (section 3.2.1.1. In this test case, first the count of entities having this inconsistency was checked and then compared with count of remaining inconsistencies after implementing the proposed solution (section 3.2.1.9). After implementing the solution, all the multiple birth date inconsistencies were removed (section 3.2.1.10). Table 4.2 shows the details of test case 1.

TABLE 4.2: Test case 1.

Test Case 1	
Error name	Multiple birth dates
Error type	Inconsistency
Evaluation tools	DBpedia Extraction Framework & Query Endpoint
Errors before solution	104
Remaining errors after solution	0
Error removal percentage	100%
Comments	All the inconsistent statements are made consistent with implemented solution.

4.1.2 Test Case 2 (Self Parents Inconsistency)

This test case deals with the self parents inconsistency (section 3.2.1.3). In this test case, first the count of entities having this inconsistency was checked and then compared with count of remaining inconsistencies after implementing the proposed solution (section 3.2.1.9). After implementing the solution, all the self parents inconsistencies were removed (section 3.2.1.10). Table 4.3 shows the details of test case 2.

TABLE 4.3: Test case 2.

Test Case 2	
Error name	Self parents
Error type	Inconsistency
Evaluation tools	DBpedia Extraction Framework & Query Endpoint
Errors before solution	95
Remaining errors after solution	0
Error removal percentage	100%
Comments	All the inconsistent statements are made consistent with implemented solution.

4.1.3 Test Case 3 (Same Values for Distinct Properties Inconsistency)

This test case deals with the same values for distinct properties inconsistency (section 3.2.1.5). In this test case, first the count of entities having this inconsistency was checked and then compared with count of remaining inconsistencies after implementing the proposed solution (section 3.2.1.9). After implementing the solution, all the same values for distinct properties inconsistencies were removed (section 3.2.1.10). Table 4.4 shows the details of test case 3.

TABLE 4.4: Test case 3.

Test Case 3	
Error name	Same values for distinct properties
Error type	Inconsistency
Evaluation tools	DBpedia Extraction Framework & Query Endpoint
Errors before solution	130
Remaining errors after solution	0
Error removal percentage	100%
Comments	All the inconsistent statements are made consistent with implemented solution.

4.1.4 Test Case 4 (Incorrect Association of Information Inconsistency)

This test case deals with the incorrect association of information inconsistency (section 3.2.1.7). In this test case, first the count of entities having this inconsistency was checked and then compared with count of remaining inconsistencies after implementing the proposed solution (section 3.2.1.9). After implementing the solution, all the incorrect association of information inconsistencies were removed (section 3.2.1.10). Table 4.5 shows the details of test case 4.

TABLE 4.5: Test case 4.

Test Case 4	
Error name	Incorrect association of information
Error type	Inconsistency
Evaluation tools	DBpedia Extraction Framework & Query Endpoint
Errors before solution	88
Remaining errors after solution	0
Error removal percentage	100%
Comments	All the inconsistent statements are made consistent with implemented solution.

4.1.5 Test Case 5 (Same Parent and Spouse Inconsistency)

This test case deals with the same parent and spouse inconsistency (section 3.2.2.1). In this test case, first the count of entities having this inconsistency was checked and then compared with count of remaining inconsistencies after implementing the proposed solution (section 3.2.2.3). After implementing the solution, more than ninety percent all the same parent and spouse inconsistencies were removed (section 3.2.2.4). Table 4.6 shows the details of test case 5.

TABLE 4.6: Test case 5.

Test Case 5	
Error name	Same parent and spouse
Error type	Inconsistency
Evaluation tools	DBpedia Extraction Framework & Query Endpoint
Errors before solution	44
Remaining errors after solution	4
Error removal percentage	90%
Comments	Three of the remaining entities are due to incorrect classification of entities by extractors.

4.1.6 Test Case 6 (Incompleteness)

This test case deals with the incompleteness (section 3.2.3.1). In this test case, first the count of entities having this inconsistency was checked and then compared with count of remaining inconsistencies after implementing the proposed solution (section 3.2.3.3). After implementing the solution, more than ninety eight percent of all the incomplete instances were completed (section 3.2.3.4). Table 4.7 shows the details of test case 6.

TABLE 4.7: Test case 6.

Test Case 6	
Error name	Incorrect entity type
Error type	Incompleteness
Evaluation tools	DBpedia Extraction Framework & Query Endpoint
Errors before solution	4885
Remaining errors after solution	72
Error removal percentage	98.5%
Comments	Seventy two remaining incomplete instances were not resolved due to extraction issues.

4.1.7 Test Case 7 (Incorrect Domain Inconsistency)

This test case deals with the incorrect domain inconsistencies (section 3.3.1). In this test case, first the count of ontology properties having this inconsistency was checked and then compared with count of remaining inconsistencies after implementing the proposed solution (section 3.3.3). After implementing the solution, more than ninety eight percent of all the incorrect domain inconsistencies were removed (section 3.3.3). Table 4.8 shows the details of test case 7.

TABLE 4.8: Test case 7.

Test Case 7	
Error name	Incorrect domain
Error type	Inconsistency
Evaluation method	Analysis of ontology
Errors before solution	4
Remaining errors after solution	0
Error removal percentage	100%
Comments	All four inconsistent domains were made consistent by changing the class hierarchy and domains.

Summary

This chapter evaluated the results of solutions provided in previous chapter. Test case 1 showed that error removal percentage of inconsistencies related to multiple birth is 100%. Test case 2 showed that error removal percentage of inconsistencies related to self parent is 100%. Test case 3 showed that error removal percentage of inconsistencies related to distinct properties is also 100%. Test case 4 showed that error removal percentage of inconsistencies related to incorrect association of information is 100% as well. Test case 5 showed that error removal percentage of inconsistencies related to same parent and spouse is 90%. Test case 6 showed that error removal percentage of incomplete instances is 98%.Test case 7 showed that error removal percentage of incorrect domain inconsistency is 100%. Figure 4.1 shows the summary of results of all experiments conducted in this research.

Errors	Total Errors	Rectified Errors	Percentage	Reason for Errors
Multiple birthdates	104	104	100%	Multiple info boxes
Self parents	95	95	100%	Multiple info boxes
Same values for distinct properties	130	130	100%	Multiple info boxes
Incorrect association of information	88	88	100%	Multiple info boxes
Same parent and spouse	44	40	90%	Regex and post processing issue
Incompleteness problem	4885	4813	98%	Missing concepts in DBpedia
Incorrect domain	4	4	100%	Ontology incorrect

FIGURE 4.1: Results summary

Chapter 5

Conclusion and Future Work

5.1 Conclusion

Effort was made to remove semantic errors from DBpedia knowledge graph which are very difficult to detect because they require real world knowledge. Most of the automatic approaches miss these type of inconsistency and incompleteness issues and also doesn't provide the reason behind these and their possible solution so this process must be done manually by humans. DBpedia is inconsistent because of very small errors which in turn have a huge impact. To improve DBpedia we proposed the approach comprising of four phases, Discovery, Analysis, Solution and Evaluation. One may get the previously discovered inconsistencies and work on that and skip the first phase. There exists three domain of errors, Ontology, mappings and extraction framework. Ontology and mappings must be improved constantly and must be reviewed by domain experts to minimize the inconsistencies. Logical errors must also be removed periodically from DBpedia extraction framework. Our Proposed approach can also be applied to other knowledge graphs. The study discovered more than 5300 errors in DBpedia knowledge graph and 98% of those errors are corrected, along with it twelve new concepts are introduced and fifteen new mappings are also created.

5.2 Future Work:

This methodology can also be applied to other knowledge graphs like Yago. Currently DBpedia is a shallow ontology and very less restrictions and features are provided for classes and properties, so DBpedia ontology could also be enriched with disjoint axioms, transitive properties, inverse properties etc so that the reasoning engine could also be able to detect inconsistencies in the ontology

Bibliography

[1] Dieter Fensel, Umutcan Şimşek, Kevin Angele, Elwin Huaman, Elias Kärle, Oleksandra Panasiuk, Ioan Toma, Jürgen Umbrich, and Alexander Wahler. Introduction: what is a knowledge graph? *Springer*, 2020.

[2] Heiko Paulheim. Knowledge graph refinement: A survey of approaches and evaluation methods. *Semantic web*, 8(3):489–508, 2017.

[3] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. Dbpedia: A nucleus for a web of open data. pages 722–735, 2007.

[4] Ontology (DBO) - DBpedia. retrieved july 22, 2021, from https://www.dbpedia.org/resources/ontology/.

[5] Diego Torres, Hala Skaf-Molli, Pascal Molli, and Alicia Díaz. Discovering wikipedia conventions using dbpedia properties. pages 115–144, 2014.

[6] Dbpedia-extraction-framework. retrieved june 22, 2021, from https://github.com/dbpedia/extraction-framework//.

[7] Gerald Töpper, Magnus Knuth, and Harald Sack. Dbpedia ontology enrichment for inconsistency detection. pages 33–40, 2012.

[8] Dbpedia profile. retrieved march 19, 2021, from https://dbpedia.org/page.

[9] Wikipedia infoboxes. retrieved march 20, 2021, from https://en.wikipedia.org/.

[10] Marcus A. Rothenberger Samir Chatterjee Ken Peffers, Tuure Tuunanen. A design science research methodology for information systems research. *Journal of Management Information Systems*, 24(3):45–77, 2007.

[11] Daniel Fleischhacker and Johanna Völker. Inductive learning of disjointness axioms. pages 680–697, 2011.

[12] Yanfang Ma, Huan Gao, Tianxing Wu, and Guilin Qi. Learning disjointness axioms with association rule mining and its application to inconsistency detection of linked data. pages 29–41, 2014.

[13] Mariano Rico, Nandana Mihindukulasooriya, Dimitris Kontokostas, Heiko Paulheim, Sebastian Hellmann, and Asunción Gómez-Pérez. Predicting incorrect mappings: a data-driven approach applied to dbpedia. pages 323–330, 2018.

[14] Daniel Caminhas, Daniel Cones, Natalie Hervieux, and Denilson Barbosa. Detecting and correcting typing errors in dbpedia. 2019.

[15] Elena Cabrio, Julien Cojan, Serena Villata, and Fabien Gandon. Argumentation-based inconsistencies detection for question-answering over dbpedia. 2013.

[16] Nouman ahmad khan. handling family relations inconsistencies in lod (dbpedia). capital university, 2020.

[17] Zhaohua Sheng, Xin Wang, Hong Shi, and Zhiyong Feng. Checking and handling inconsistency of dbpedia. pages 480–488, 2012.

[18] Túlio Martins and Julio Cesar dos Reis. Mechanism for inconsistency correction in the dbpedia live. 2019.

[19] Piyawat Lertvittayakumjorn, Natthawut Kertkeidkachorn, and Ryutaro Ichise. Correcting range violation errors in dbpedia. 2017.

[20] Jens Lehmann and Lorenz Bühmann. Ore-a tool for repairing and enriching knowledge bases. pages 177–193, 2010.

[21] Mark Musen. The protégé project. *AI Matters*, 1(3):45–77, 2015.

[22] Peter Exner and Pierre Nugues. Entity extraction: From unstructured text to dbpedia rdf triples. pages 58–69, 2012.

[23] Dustin Lange, Christoph Böhm, and Felix Naumann. Extracting structured information from wikipedia articles to populate infoboxes. pages 1661–1664, 2010.

[24] Jiaoyan Chen, Xi Chen, Ian Horrocks, Erik B. Myklebust, and Ernesto Jimenez-Ruiz. Correcting knowledge base assertions. pages 1537–1547, 2020.

[25] Benno Kruit, Peter Boncz, and Jacopo Urbani. Extracting novel facts from tables for knowledge graph completion. *Springer*, pages 364–381, 2019.

[26] Baoxu Shi and Tim Weninger. Proje: Embedding projection for knowledge graph completion. 31(1), 2017.

[27] Meihong Wang, Linling Qiu, and Xiaoli Wang. A survey on knowledge graph embeddings for link prediction. *Symmetry*, 13(3):485, 2021.