

M.Sc. Engg. Thesis

**Wikipedia Entry Augmentation by Enriching
Source Data Set Using a Multi-lingual Ontology
Based Framework**

By
Md. Tasnim Manzur Ankon

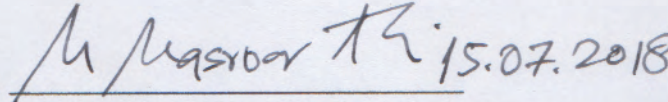
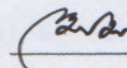
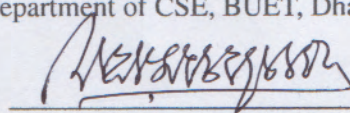
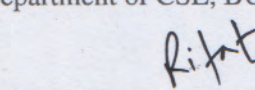

Submitted to
Department of Computer Science and Engineering
in partial fulfilment of the requirements for the degree of
Master of Science in Computer Science and Engineering

Department of Computer Science and Engineering
Bangladesh University of Engineering and Technology (BUET)
Dhaka-1205

July, 2018

The thesis titled “**Wikipedia Entry Augmentation by Enriching Source Data Set Using a Multi-lingual Ontology Based Framework**”, submitted by Md. Tasnim Manzur Ankon, Roll No. 1015052071, Session October 2015, to the Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology, has been accepted as satisfactory in partial fulfilment of the requirements for the degree of Master of Science in Computer Science and Engineering and approved as to its style and contents. Examination held on July 15, 2018.

Board of Examiners

1.  15.07.2018
Dr. Muhammad Masroor Ali
Professor
Department of CSE, BUET, Dhaka.
(Supervisor) Chairman
2. 
Dr. Md. Mostofa Akbar
Head
Department of CSE, BUET, Dhaka. Member
(Ex-Officio)
3. 
Dr. M. Kaykobad
Professor
Department of CSE, BUET, Dhaka. Member
4. 
Dr. Rifat Shahriyar
Assistant Professor
Department of CSE, BUET, Dhaka. Member
5. 
Dr. Kazi Muheymin-U-Sakib
Professor
IIT, University of Dhaka, Dhaka-1000 Member
(External)

Candidate's Declaration

This is to certify that the work presented in this thesis titled “**Wikipedia Entry Augmentation by Enriching Source Data Set Using a Multi-lingual Ontology Based Framework**” is the outcome of the investigation carried out by me under the supervision of Professor Dr. Muhammad Masroor Ali in the Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology (BUET), Dhaka. It is also declared that neither this thesis nor any part thereof has been submitted or is being currently submitted anywhere else for the award of any degree or diploma.

Md. Tasnim Manzur Ankon
Candidate

Contents

<i>Board of Examiners</i>	i
<i>Candidate's Declaration</i>	ii
List of Figures	vi
List of Tables	vii
List of Abbreviations	viii
Acknowledgements	ix
Abstract	x
1 Introduction	2
1.1 Wikipedia Entry Augmentation	2
1.2 Objectives with Specific Aims	4
1.3 Summary of Contributions	4
1.4 Thesis Organization	5
2 Related Work and Current Status	6
2.1 Present Status of Wikipedia Entries	8
2.2 Mapping of Multiple Languages	8
2.2.1 Translation	9
2.2.2 Structural Mapping	9
2.3 Live Extraction of Structured Data	11
2.4 Search Engine for DBpedia	12
2.5 Augmentation of a Feature Set Using Linked Open Data	12

2.6	Matching HTML Tables to DBpedia	12
2.7	Depiction of Entities of Different Languages	15
2.8	Analytical Summary	17
2.9	Research Questions	18
3	Preliminaries	19
3.1	Basic Terminology	19
3.1.1	Semantic Web	19
3.1.2	Ontology	21
3.1.3	Concepts	21
3.1.4	Entities	22
3.1.5	Properties	22
3.1.6	Relationships	22
3.1.7	Ontology Mapping	23
3.1.8	Multilingual Ontology	23
3.1.9	Resource Description Framework	23
3.1.10	Wikipedia	24
3.1.11	DBpedia	25
3.1.12	Date Set	26
3.2	Main Challenges in Focus	26
4	Methodology	30
4.1	Proposed Methodology	30
4.2	Framework for Wikipedia Entry Augmentation	30
4.2.1	Multilingual Ontology	31
4.2.2	Multilingual Ontology Mapping	32
4.2.3	Information Retrieval	34
4.2.4	Data Sets of the Ontology	35
4.3	Submerging Multilingual Entities	35
4.3.1	Language Selection	36
4.3.2	Translation and Mapping With DBpedia	36
4.3.3	Extracting Knowledge Base	38
4.3.4	Preparing Data Sets	38

4.3.5	Set of Semantic Rules	39
4.3.6	Complexity Assessment	41
4.3.7	Finalizing New Data Set	42
4.4	Summary	42
5	Experiments	44
5.1	Competency of the System	44
5.2	Selection of Language and Translation	45
5.3	Populating Data Sets	45
5.4	Processing the Data Sets	46
5.5	Application of Semantic Rules	46
5.6	Submerging Data Set	47
5.7	Preparing the Export File	47
5.8	Experimental Result	52
6	Conclusion and Future Work	54
6.1	Compendium of Attainments	54
6.2	Integration of New Languages	55
6.3	Scope for Bengali DBpedia	55
6.4	Gateway to Many More	57
A	Bibliography	58
B	Source Code	63
B.1	System to Submerge Entities Based on Multilingual Ontology	63

List of Figures

2.1	The process of DBpedia extraction framework.	10
3.1	Semantic Web stack	20
3.2	A small snippet from an ontology.	21
3.3	A simple example of a concept or class.	22
3.4	A simple entity defining Cricket.	22
3.5	Structure of a triple format.	24
3.6	The official logo of DBpedia.	25
3.7	A sample Data Set.	26
4.1	Generation of combined ontology.	33
4.2	Linking dictionary to ontology.	33
4.3	The process of submerging multilingual entities.	37
5.1	The parent tag as found in the generated XML file.	48
5.2	The subject-property-statement format represented in the exported XML file when opened in Mozilla Firefox.	48
6.1	A sample English Wikipedia page.	56
6.2	Bengali Wikipedia counter part to that of Figure 6.1	56

List of Tables

2.1	State of English and Bengali languages for which mappings exist in the DBpedia mapping wiki.	7
2.2	Data set statistics for DBpedia 2014 with the data set statistic for DBpedia 2015-04. See Table 2.1 for legends.	7
2.3	Comparison among the nine most spoken languages with respect to the available resources in DBpedia	14
2.4	Comparison among entities, properties and statements of English and French language in DBpedia	14
5.1	Experimental results obtained by implementing the process of sub-merging multilingual entities in case of French language.	50
5.2	Experimental results obtained by implementing the process of sub-merging multilingual entities in case of French language.	51
5.3	Proposed performance metrics for executed system.	53

List of Abbreviations

GATE General Architecture for Text Engineering

JAPE Java Annotations Patterns Engine

ME Maximum Entropy

NER Named Entity Recognition

OWL Web Ontology Language

RDF Resource Description Framework

SVM Support Vector Machine

TF-IDF Term Frequency- Inverse Document Frequency

Acknowledgements

First of all, I would like to declare that all the appraisals belong to the Almighty **ALLAH**.

I would like to thank my supervisor Professor Dr. Muhammad Masroor Ali for introducing me to the fascinating and prospective field of semantic web and wikipedia augmentation. I have learned from him how to carry on a research work, how to write, speak and present well. My heartiest gratitude goes to him for his patience in reviewing my so many inferior drafts, for correcting my proofs and language, suggesting new ways of thinking, leading to the right way, and motivating me to continue my research work. I again express my indebtedness, sincere gratitude and profound respect to him for his constant supervision, suggestions and whole hearted guidance throughout the progress of this work.

I convey my heartfelt reverence to my parents and other family members for giving their best support throughout my work to overcome the tedium of repetitive trials to new findings.

Finally, every honor and every victory on earth is due to ALLAH, descended from Him and must be ascribed to Him. He has endowed me with good health and with the capability to complete this work. I deeply express my sincere gratitude to the endless kindness of ALLAH.

Abstract

The domain of traditional web is gradually evolving with the adaptation of newer techniques, which includes Semantic Web. Integration of web content using ontologies in a language independent manner is a required feature in this process. For better utilization of the resources, it is necessary that the ontology, which is working as a central knowledge repository, to be language independent as well. Apart from being language independent, we should consider that the better the source data set of the ontology, the better options there will be for the adaptation of new knowledge.

In this thesis, we introduce a framework for multilingual ontology, which is be able to adapt to the addition of new languages, as well as the addition of new data to the existing sources. The framework, which itself is an extension of the framework used at present, augments the domain of the current ontology. The augmentation is ensured by introducing a universal technique to integrate infinite number of languages to the understanding of the multilingual ontology. Once elaboration of the framework is done, we highlight the significance of efficient data extraction techniques from the ontology.

This thesis also introduces a way to improve the extraction technique by concentrating multiple data sources into one single source. We present a process where machine-readable properties of individual entities are filtered through intelligent techniques and a precise knowledge source is generated.

Thus, sub-merging multiple knowledge bases into one single and richer data set. Lastly, we present the results obtained by experimental implementation of the sub-merging mechanism to demonstrate the magnitude of enhancement and its contribution to fulfill our ultimate goal to augment Wikipedia entries.

Publications

- [1] M. T. M. Ankon, S. N. Tumpa and M. M. Ali, “A Multilingual Ontology Based Framework for Wikipedia Entry Augmentation,” In *19th International Conference on Computer and Information Technology (ICCIT)*, pp. 541-545, IEEE, 2016, held at North South University, Dhaka, Bangladesh.
- [2] M. T. M. Ankon and M. M. Ali, “Wikipedia Entry Augmentation by Sub-merging Entities Based on Multilingual Ontology,” In *6th International Conference on Informatics, Electronics & Vision (ICIEV) & 7th International Symposium in Computational Medical and Health Technology (ISCMHT)*, IEEE, 2017, held at University of Hyogo, Hyogo, Japan.

Chapter 1

Introduction

1.1 Wikipedia Entry Augmentation

The World Wide Web acts as a store for immeasurable amount of data. Each and every day this amount keeps on increasing. All software and web systems that use such web content as its data source are thus, having to adapt with the increasing amount of information. However, such software and systems face much difficulty when it has to traverse through the immense amount of unstructured information. Hence, the concept of bringing the entire information set under a central knowledge repository got significant. Any software or web solution that may require the use of the content in web can be assisted by the central knowledge repository. A result of which is that, the entire world of artificial intelligence will be able to understand all the information in the web. Thus, the concept mentioned earlier, got its own research area in the domain of Semantic Web [1–3].

Semantic web, with the motive to make the entire set of web content machine readable, has adapted the usage of ontologies. These ontologies describe the concepts that can define a specific entity. However, a single entity can be defined in a number of ways, a probable reason of which is the adaptation of multiple languages. This adaptation brought forth the concept of making the central knowledge repository language independent [4].

In order to identify any entity by a machine reading mechanism, there is the requirement for a structured definition of that entity. Since the web is already flooded with diverse information, it was necessary to take an existing information set, like an encyclopedia, to be the fundamental

source for the structured information. Wikipedia, a free encyclopedia, acted as the perfect source of information. Semantic web took a huge step when the structured version of Wikipedia came into being. Researchers termed the version as DBpedia [5].

Wikipedia, being an immense knowledge repository itself, is suitable for human eyes. It is up to the human to read and search for required data. When it comes to a machine, Wikipedia does not provide a machine friendly format. Hence, the unstructured data of Wikipedia was made into a structured data set with DBpedia project. DBpedia itself hosts individual page corresponding to each page of Wikipedia. However, the difference lies in the fact that the DBpedia page simply demonstrates the entire information set in a list of property-value or property-statement pairs. Such pairs, though not friendly for human eyes, are perfectly suitable for machine reading. The pages or entities are denoted as the subject and the property-statement pairs are its description [1].

DBpedia contains Uniform Resource Identifiers (URI) against each possible entity. All the properties that define the entity are identified by a derivation of the initial URI [5]. Basically, DBpedia is the version adapted from Wikipedia, for Semantic Web. When the web received the ability to adapt multiple languages, it became necessary for DBpedia to be able to adapt multiple languages as well. Here arose the requirement for a multilingual ontology [6]. The multilingual ontology will require an information set that has the elements of all available languages mapped [4] and ensure that the ontology is able to adapt, without any major issues, the addition of new languages [7].

One of the major purposes of the ontology is to map the exact definition and description for all the content in the web [8]. It acts as a directory which can direct towards the actual description of an entity. When we talk about the description, we represent the concept of a source object. Such sources act as the data repository for the ontology. It can be noted that the page denoting the properties of any entity in DBpedia is a source for the ontology. So, if the data in these sources are structured, the efficient the ontology will be [9, 10]. As mentioned in this section, the immense amount of unstructured data acts as the main obstacle for intelligent systems [11]. Therefore, the ontology must have to abide by a particular framework, which restricts the type of structure readable by the ontology. This framework provides the generalized structure that acts as a guideline for anyone who intends to bring the knowledge in web into the domain of AI's understanding [4, 7].

Our research focused on establishing a framework that integrates the subject-property-statement triple structure of DBpedia into the domain of understanding of the multilingual ontology [7]. With the help of this particular structure, we propose a system that is able to combine the knowledge base of multiple languages and create a rich and enhanced data source [12]. Apart from serving its main purpose, the system provides a better option for addition of new languages to the DBpedia as we know it. Our ultimate goal of augmenting Wikipedia was served when we implemented the system in a user-friendly manner. We also demonstrate the magnitude of enhancement resulted by the implementation of our system through suitable examples.

1.2 Objectives with Specific Aims

The objectives of this thesis are enumerated below:

- Hypothesizing a framework for the multilingual ontology.
- Establish a generalized system that can create language independent data sets, aligned with the framework proposed for multilingual ontology.
- Generate single, rich and extract worthy data set by ensuring the best of quality through the formulation and implementation of a set of semantic rules.
- Augment Wikipedia entries with the help of the generated data sets.

1.3 Summary of Contributions

The main contributions of the thesis are as follows:

- A framework that is able to handle adaptations due to update of existing resources and addition of new language to the existing multilingual ontology.
- A web based system that can prepare a rich and efficient data set by merging resources from the resources of two languages.
- A set of semantic rules that ensures that the generated data set are of highest efficiency.

- The passive motive of giving a strong base to any new language that is to be brought under the domain of the multilingual ontology.

1.4 Thesis Organization

The remainder of the thesis is organized as follows: Chapter 2 introduces mapping and extraction techniques used for DBpedia and some handpicked works related to them. Chapter 3 defines the terminologies and notations used in this thesis. Chapter 4 presents the entire concept that establishes the framework and the methodology that is used to enhance the data set of DBpedia. Chapter 5 reports experimental results on real world data sets. Finally, Chapter 6 enumerates the attainments of this thesis and then concludes the thesis suggesting possible future extensions.

Chapter 2

Related Work and Current Status

In recent years, enhancement of DBpedia has been a major focus in the field of Semantic Web, basically through the adaptation of multiple languages [4]. However, quite a lot of research has been conducted on improving the extraction process, which has been one of the most popular sections of interest [13] within the domain of discussion. We have selected a few of the extraction techniques which are somewhat dependent on the knowledge base of DBpedia. We present these techniques to demonstrate the significance of improving the framework that enables multilingual data extraction and the quality of the data set itself.

As we get introduced with the techniques, we will also elaborate the current status of the Wikipedia entries, which are the actual data set for consideration when it comes to data extraction from DBpedia [1]. A further elaboration of these extraction techniques will be supported by our presentation of mapping techniques currently used between Wikipedia and DBpedia.

Table 2.1: State of English and Bengali languages for which mappings exist in the DBpedia mapping wiki.

Language	Instances, LD, all	Instances, CD, all	Raw Properties, CD	Mapping Properties, CD	Raw State-ments, CD	Mapping State-ments, CD	Type State-ments, CD
en	4,806,150	4,563,644	58,781	1,354	73,627,718	67,054,254	35,361,157
bn	33,915	474	7,532	54	346,442	5,656	3,481

Here, LD means localized data sets, CD stands for canonicalized data sets and MD means number of instances for which mapping-based infobox data exists according to [14].

Table 2.2: Data set statistics for DBpedia 2014 with the data set statistic for DBpedia 2015-04. See Table 2.1 for legends.

	Instances, CD, all	Instances, CD, with MD	Instances, LD, all	Mapping Properties, CD	Raw Properties, CD	Mapping State-ments, CD	Raw State-ments, CD	Type State-ments, CD
2014	26,136	2,160	29,631	83	6,609	30,350	271,070	19,015
2015-04	29,729	474	33,915	54	7,532	5,656	346,442	3,481
%	+14%	-78%	+14%	-35%	+14%	-81%	+28%	-82%

2.1 Present Status of Wikipedia Entries

As Wikipedia has separate editions in different languages, same entity has been described in multiple pages of different editions of Wikipedia. For example, the instance “Rice” has been described as “চাল” in Bengali Wikipedia, which are generally linked with each other by the cross language linking section of Wikipedia [15].

After extracting data from Wikipedia pages, generally DBpedia originates two types of data set. One is localized data set and the other is canonicalized data set. The localized data sets include all the entities which are described in a particular human language and the entities can be determined with URIs of that particular language. The canonicalized data sets comprises entities which have an equivalent page in the English version of Wikipedia and the same entity is determined with the identical URI from the umbrella namespace, <http://dbpedia.org/resource/> [14].

Table 2.1 states the number of instances in both data sets broadly, raw-infobox and mapping properties, raw-infobox and mapping statements and type statements for English and Bengali languages. The table covers only those data for which DBpedia mapping contains Wikipedia mappings [14]. Table 2.2 presents the data set statistics for DBpedia 2014 with the version 2015-04 which allows comparison between these two versions.

2.2 Mapping of Multiple Languages

To describe the idea of mapping multiple languages, we are going to use an example. Let us again consider the English word “Rice”. Any particular mention of this word in any web content should be linked to a single source. As mentioned previously, the Bengali translation of “Rice” is “চাল”. Any mention of this Bengali word (চাল) in any web content should also be linked up to that same identifier. This universality should be maintained for all the languages. A major focus of this thesis is to find out a generalized way to augment the vocabulary, by integrating multiple languages. The vocabulary acts as the data set for the multilingual ontology. As we move forward, some concepts regarding the augmentation and extraction will be presented.

2.2.1 Translation

A way to map the English data set to Bengali data set can be direct translation. In such a case, the words in English dictionary, which acts as the data set for mapping with the existing ontology, is directly translated to form a Bengali dictionary. After that, the mapping is extended from the English word to Bengali. The way does seem very straightforward.

For every straightforward way, there will be a number of disadvantages. Most significant disadvantage in this case is, none of the web contents are structured. As discussed earlier in this paper, for a single ontology, the data needs to be in a universal format, recognized by all search engines and any other artificial intelligence tool. The contradiction of this particular requirement, thus, brings forward the major disadvantage of direct translation.

The structured data formatting in Wikipedia [5] has been one of the major advantages during the formulation of DBpedia. In case of direct translation, the advantage of structured data format is completely ignored. Hence, this alone is enough to depict the procedure, on its own as naive.

2.2.2 Structural Mapping

The creation of DBpedia from Wikipedia ensured a standard way of mapping. This procedure is discussed here in brief. An elaborate description can be studied from [5].

DBpedia is a project which extracts information from Wikipedia pages. Generally, Wikipedia articles mostly consist of free format text but, it also includes structured information along with the free format text in the articles with the help of infobox, categorization information, images, external web page links etc. An extraction framework is necessary to extract this structured information from Wikipedia, which helps to turn the wiki data into a large knowledge base for DBpedia. Conceptually, a general architecture of the extraction framework can be described as follows.

There is an input section where Wikipedia pages are read from external source using dump or API. This is just like a window to make the entrance for the Wikipedia pages. After that, a parsing technique is applied which transforms the Wikipedia pages into a syntax tree by determining data types, converting values etc. Wiki parser is mostly used here. There are some extractors, notably labels, abstracts, geo-coordinates, images and infoboxes, which are used to extract labels, abstracts, geographical coordinates and so on. The outputs of these extractors are

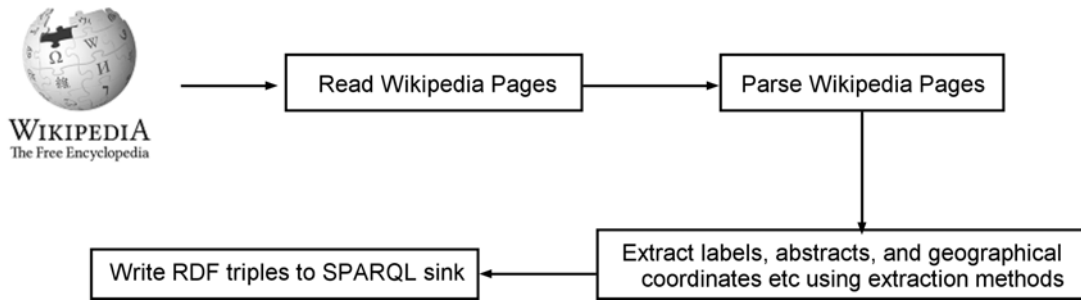


Figure 2.1: The process of DBpedia extraction framework.

a set of Resource Description Framework (RDF) statements which are written to a dump sink or SPARQL sink.

Usually DBpedia extracts information from the meta data of infoboxes of Wikipedia. According to Wikipedia, “An infobox is a fixed-format table designed to be added to the top right-hand corner of articles to consistently present a summary of some unifying aspect that the articles share and sometimes to improve navigation to other interrelated articles. Many infoboxes also emit structured meta data which is used by DBpedia and others” [16].

There are various extraction processes for translating Wikipedia pages to RDF statements like raw infobox extraction, mapping based infobox extraction, feature extraction, statistical extraction etc [5]. The raw infobox extraction and mapping based infobox extraction are most commonly used to extract data from Wikipedia.

Raw Infobox Extraction

The raw infobox extractor directly maps entities and properties from Wikipedia infobox to RDF statements without normalizing it. It only produces language specific entities and properties using the term used in the Wikipedia infoboxes. Normally, a generic heuristic based algorithm is used for the extraction process [17]. The extracted data obtained from this process is useful when a particular infobox is still unmapped in the DBpedia Mapping Wiki¹.

But the problem is,

- All the infobox templates for same category of entities are not same. Different group of Wikipedia editors use different templates to create Wikipedia pages on same type of entities. For this reason, same property has been termed differently in different templates

¹<http://mappings.dbpedia.org>

like “name” and “title” both can be used for the information regarding the title of a book.

- The resource of extracted output may not belong to a class in the DBpedia ontology as it extracts data from all infoboxes.
- The quality of extracted data is lower as it directly extracts data without using any other external knowledge.

Mapping-based infobox extraction has been introduced to solve these problems.

Mapping-based Infobox Extraction

The mapping-based infobox extractor extracts data only from the infoboxes for which there exist mappings from Wikipedia infoboxes to DBpedia ontology in the DBpedia mapping wiki. The mapping based infobox extraction provides manual mapping from Wikipedia infoboxes to DBpedia ontology. In this process, attributes of infoboxes are mapped to the corresponding properties in the DBpedia ontology. Mapping based extraction makes the description of information more standard in the DBpedia knowledge base. The problem of using different infobox templates for the same entity and different attribute names for the same property can be removed by this extraction. The quality of extracted data can be increased by assigning specific data types to the values.

2.3 Live Extraction of Structured Data

The data extraction process for DBpedia from Wikipedia has been stated as quite an important aspect in the field of Semantic Web [1]. However, recent researches have found that the process of extraction is quite heavy in the matters of resource consumption and is of lower efficiency in case of fetching the latest data in Wikipedia [13]. Based on this issue, a good amount of research have been conducted to establish a better extraction mechanism.

Live extraction of structured data is an extraction process which is concerned with the data added to Wikipedia in real time. The technique processes the data on the fly, creating high level RDF files and enhancing the ontology whenever required [13]. The DBpedia-Live framework focuses on timely synchronization of DBpedia with Wikipedia. Thus, ensuring up-to-date information at all times.

2.4 Search Engine for DBpedia

As DBpedia is widely used by a vast domain of people across the world, the means to provide a way to visualize the data became significant. SPARQL was introduced as the query language in this regard [1]. However, SPARQL is basically usable by people with enough knowledge on structured query languages. Without the basics, it is quite difficult to use. Hence, the data extraction mechanism is in need of a user-friendly process.

A research to meet the above mentioned need brought up the concept and implementation of a search engine that focuses on knowledge extraction from DBpedia. Entitling the system as DBpediaSearch, the system allows users to provide SPARQL queries without the restriction on human understandable keywords [18]. High level knowledge and search result filtering allow the user to get the taste of generalized search engines. With the human interaction enabled, the extraction process is bound to take a great leap.

2.5 Augmentation of a Feature Set Using Linked Open Data

As the world wide web is loaded with unstructured data [10], a good number of researchers are focusing on bringing the knowledge base of a limited number of sources under a definite structure at a time. A recent study brought in the knowledge base from the review site *MovieTweatings* under a definite structure. This extracted knowledge base was added to the domain of the multilingual ontology with the motive to augment its knowledge base [19]. The unlimited number of structures followed by the data in the web can easily help us conclude that the major knowledge base is unstructured. Hence, initiatives are to be taken to bring more and more data within the domain of understanding of the multilingual ontology, even if its one knowledge ground at a time.

2.6 Matching HTML Tables to DBpedia

With the enhancement of software and web development technologies, the programming languages have been upgraded to a great extent. With time, the languages have become more user-friendly and easily understandable. As such, the output of the languages also provides

structured visualization. The introduction of advanced level markup languages has a particular bit of contribution in this regard.

The knowledge represented by high end markup languages, let's take HTML for example, provides a structured visualization of the data. Concentrating an HTML table, where each cell denotes a particular set of knowledge that can be linked up by its accommodating row and column, provides a clear idea to both the user and the AI underlying behind it. Recent researches have provided with ample amount of progress towards recognizing the knowledge represented by such HTML tables. Significant achievement was accomplished once a system was established to link up this knowledge base with DBpedia [20]. The contribution brought forth the concept that, not only the markup languages, but also any programming language can be used to augment the knowledge base of DBpedia. Hence, it came to our understanding that the more we enrich the source data set of DBpedia, the better will systems mentioned above perform.

Table 2.3: Comparison among the nine most spoken languages [21, 22] with respect to the available resources in DBpedia [23].

Language	Localized Instances	Canonicalized Instances	Mapping based Properties	Mapping based Statements	Raw Infobox Properties	Raw Infobox Statements	Type State-ments
Arabic (ar)	396,695	263,345	279	226,054	463	245,974	485,625
Bengali (bn)	40,025	34,092	92	18,431	92	19,951	31,440
English (en)	4,678,230	4,678,230	1,379	37,549,405	2,062	30,024,092	36,704,825
Spanish (es)	1,120,144	731,184	546	3,936,855	868	4,028,061	4,722,765
Hindi (hi)	100,303	43,520	0	0	0	0	0
Japanese (ja)	958,534	422,183	418	1,311,093	748	1,328,514	1,395,299
Portuguese (pt)	865,889	597,909	605	3,219,540	1,131	3,222,469	3,109,737
Russian (ru)	1,114,029	618,319	153	1,964,313	205	1,912,672	2,449,185
Chinese (zh)	784,931	430,425	0	0	0	0	0

Table 2.4: Comparison among entities, properties and statements of English and French language in DBpedia [23].

Language	Localized Instances	Canonicalized Instances	Mapping based Properties	Mapping based Statements	Raw Infobox Properties	Raw Infobox Statements	Type State-ments
English (en)	4,678,230	4,678,230	1,379	37,549,405	2,062	30,024,092	36,704,825
French (fr)	1,591,318	1,027,815	933	4,407,915	1,363	5,704,950	5,020,942

2.7 Depiction of Entities of Different Languages

Wikipedia initially started with only English language in its stock. As a result, English entries in Wikipedia still hold the highest amount of data among all other languages [24]. From its origination in 2001 till now, Wikipedia has successfully integrated about 290 languages and is still planning on adding more [24]. But the problem still persists that only a handful of languages have enough entities, which can be designated as rich. Only about 15 languages have crossed the million-entry mark until now [25]. All other languages are still trying to cope up with the amount of data in English language. Table 2.3 portrays a simple scenario by providing a comparison on the basis of the present statistics of the localized and canonicalized data sets of the nine most spoken languages [21, 22].

As mentioned earlier, the data from Wikipedia is extracted and brought under a structured and machine-readable format by DBpedia [5]. With this knowledge in hand, it can be stated that DBpedia generates some sort of data set after extracting the data from Wikipedia. To be more specific, DBpedia generates two types of data sets. The first one is a localized data set, which simply specifies all the entities of a particular language with no connection or reference to any other language. These entities can only be accessed by the local URIs of that language. The second one is a canonicalized data set. The entities in this data set has equivalent pages in English Wikipedia and can be directly accessed by similar URIs, starting with the generic namespace of DBpedia, <http://dbpedia.org/resource/> [23], followed by the simple derivation that denotes the particular entity.

Among the nine most spoken languages, none other than the entities of English languages has all the entities linked up with a unique URI. A major percentage of the other eight languages are completely dependent on the specific language based URI. This means that even though there may be a page of a particular entity in multiple languages, they will still be identified by separate URIs, based on the language. A simple example may be able to elaborate further. The page that denotes “Earth” in Spanish DBpedia is <http://es.dbpedia.org/page/Earth>. One of the properties of this page is <http://es.dbpedia.org/property/origen>, that can be translated to “origin” property in English. So, when the same property will be referenced through the Spanish DBpedia page, the URI will be derived from <http://es.dbpedia.org>. In case the statement corresponding to this property is same, there

will be repetition. In case the statement is different, merging the two pairs and referencing them from the same URI would have been able to augment the knowledge resource. The lack of mapping refrains us from doing so and this becomes a significant proof of the contradiction to the statement that all entities in the world of web should have a unique identifier of its own. However, this part only elaborates the difference of entities when considering the instances. The difference increases exponentially when we consider the properties. The example mentioned describes the difference with respect to one property in a single page. There are various cases of the same property being referenced through the URI of the entity. Hence, the “origin” property will have a tonne of other descriptions when we consider up to the property level.

Table 2.3 also states the differences in properties that can be linked among multiple languages. To understand the extraction techniques, we need to have a general idea about infoboxes, which has been described in Section 2.2.2 of this thesis. Infoboxes can be defined as a particular type of source, which contains extractable data relevant to the page that hosts one or more infoboxes. The data in infoboxes may redirect to resources that better define the instance of the page, or can be used by other systems, an example of which is DBpedia. The data from these infoboxes are extracted using two extractors [1, 23], which has already been introduced in Section 2.2.2 of this thesis. The mapping-based extraction process extracts data only from the infoboxes which has a language-specific extraction mapping to the DBpedia ontology. Data extracted from here are usually referenced by a single identifier, irrespective of the language. The raw infobox extractor on the other hand, extracts data from all the infoboxes. Data extracted in this way contains both mapped and not mapped properties and statements.

To describe further, let us consider a scenario. A single entity in a language will have quite a few properties. A single property can have multiple definitions, i.e. statements. Hence, if the data is not mapped and are scattered in case of multiple languages, the identifiers will create a huge problem. When we consider multiple languages, a jumble of identifiers will be presented for a single property of a particular entity. It should also be noted that the extraction process has been conducted on only 127 languages, among the whole gamut of languages included in Wikipedia [23]. When all the 290 languages that are included in Wikipedia [24] will be brought under the extraction mechanism, it will be possible to make the majority portion of the web content machine readable.

A brief comparison between the entries of English and French languages are presented in

Table 2.4. This thesis, whilst focusing on enhancing the data set of DBpedia, will be working with the data sets of English and French.

2.8 Analytical Summary

The following can be observed concerning the above approaches:

- Based on the researches presented above, it can be easily concluded that quite a bit of work has been conducted to improve the process of data extraction from DBpedia.
- Almost all of the extraction processes fetch the knowledge from the ontology. Hence, the structure of the ontology has a great significance in the operational complexity of each of the system.
- The concept used in the current mapping system itself is a strong one. Issues are stated regarding the operation time and fetching of the latest knowledge. Even in this case, the mapping is conducted based on the triples stated in the multilingual ontology.
- Researchers are working on bringing up more and more aspects that can help revolutionize the process of mapping data across the internet in a structured manner. All these mapping techniques are dependent on the multilingual ontology.
- Search engines and markup language mappers are itself acting as an engine that can map data with the ontology in an easier possible way.
- The ontology itself is dependent on the data set, which acts as the dictionary. The triples in the ontology are direct representations from the set of data obtained from these dictionaries.
- The better the structure of the ontology and the more optimized the data set is, the better will each of the mapping techniques be and the better will the intelligent systems that are presented here, perform.

2.9 Research Questions

From the previous studies, we can formulate the following issues as our research questions to be addressed in this thesis.

Framework of the Ontology: As almost all of the systems described in this thesis are dependent on the ontology itself, it is quite clear the structure or framework of the ontology should provide all systems with the easiest possible way of extracting knowledge.

Adaptation of New Features: The ontology, when rich in knowledge, will be able to provide all such systems with more and more features. With the introduction of the concept of language independent web, the adaptation of multiple languages has become a major requirement. Hence, the ontology should be able to adapt to changes and addition of new knowledge in a way that creates the least amount of ripples.

Efficiency of the Data Sets: The triples in the ontology are formed with the help from the data sets for each particular language, when it comes to DBpedia. Thus, with the data sets becoming more efficient, the ultimate result will affect the ontology, in a way that the ontology becomes richer exponentially.

Chapter 3

Preliminaries

3.1 Basic Terminology

Our thesis establishes the framework for a language independent ontology and develops a system that can enhance the data source for the said ontology. In this section, we introduce some basic terminologies, and intend to cover up the basic understanding of the entire domain, which includes ontology mapping, data sets, RDFs and so on. We also elaborate about the implementation of ontology mapping in Wikipedia and DBpedia in this section. Thus, at the end of this chapter, we will be prepared to get a general understanding of the entire study.

3.1.1 Semantic Web

Semantic Web [1, 26, 27] is one of the major and significant branches of technology that has a distinguishable contribution in various fields relevant to machine learning, artificial intelligence and so on. It is simply defined as structuring the entire knowledge base in the traditional web in a machine readable format. An understandable definition is that, it establishes a knowledge repository for the artificially intelligent systems in a manner human beings sees and understands the present knowledge base through websites and web systems.

The idea of Semantic Web itself is quite old. The inventor of World Wide Web, Tim Berners Lee, mentioned the idea of a knowledge repository that has the capability of analyzing all the data in the web [28]. The concept was formally introduced in 2001, stating as the future of web, which will be working as an extension to the web and simplifying the web system that

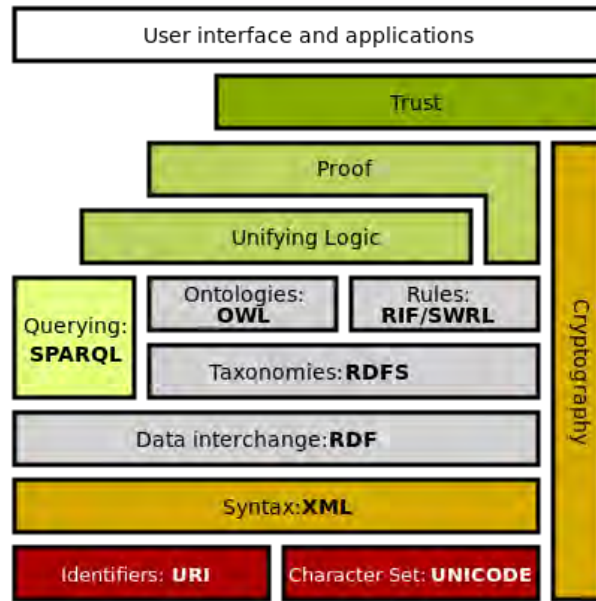


Figure 3.1: Semantic Web stack [32].

was prevailing at the time [29]. The idea has gradually evolved to a reality and as of 2013, more than 4 million web domains contained semantic web markup [30].

An idea of a special agent is quite popular when explaining Semantic Web. This agent is basically an intelligent system that has the capability to walk around the web and extract as much information about any particular entity. The agent can only extract meaningful data from the web regarding the particular entity. Let's add here that the meaning of the word *semantic* is "meaning". Thus, Semantic Web is a separate kind of web, built on top of the current or traditional web that is capable of providing meaningful data to the agent mentioned. It does so by adding a brand new layer on top of the traditional web. The main functionality of this layer is to add machine-understandable meanings to the traditional web [1].

Semantic Web [1, 26, 27], itself being a layer on the traditional web, again works through multiple layers. Figure 3.1 represents the layered implementation of semantic web. The bottom layer being a raw XML type, is supported by a layer of RDF triples [1–3, 31]. A supporting layer of RDF Schema layer [1–3, 31] helps to generate the ontologies. Ontology layer provides structured information, which is then mapped by Logic layer that declares the relevant knowledge. The Proof layer and Trust layer validates the knowledge and generates digital signature respectively.

```
<rdf:RDF
  xml:base="http://dbpedia.org/ontology/"
  xmlns:foaf="http://xmlns.com/foaf/0.1/"
  xmlns:owl="http://www.w3.org/2002/07/owl#"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns="http://dbpedia.org/ontology/">

  <owl:Ontology rdf:about="http://dbpedia.org/ontology/">
    <rdf:type rdf:resource=
      "http://purl.org/vocommons/voaf#Vocabulary"/>
  </owl:Ontology>

</rdf:RDF>
```

Figure 3.2: A small snippet from an ontology.

3.1.2 Ontology

Ontology is the formal definition of a common set of terms that describes a particular entity. It hosts the terms that are used to define and represent a certain area of knowledge [1]. Its conceptual structures provide the key to machine-processable data [33]. The terms or concepts define the particular domain and links it up with multiple dictionaries to elaborate the knowledge further. Knowledge engineers can manually create ontologies, or certain techniques can be used through information extraction to create one. Figure 3.2 represents a small section of an ontology.

3.1.3 Concepts

Concepts, represented by the keyword “Class”, acts as the main building block for an ontology. Any entity that shares characteristics with other entities, all belong to the same concept or class. Considering an example, all sports are part of the class entertainment. A class can have multiple sub-classes. Here, sport is the sub-class of entertainment. Entertainment itself can have other sub-classes. A certain domain in the class of entertainment defines sports further. Hence, a sub-class categorizes a specific domain of entertainment. Figure 3.3 defines a simple class where all sports belong to the class Entertainment.


```
<owl:Class rdf:Id="Sport "  
    <rdfs:subClassOf rdf:resource="#Entertainment ">  
</owl:Class>
```

Figure 3.3: A simple example of a concept or class.

```
<owl:Thing rdf:Id="Cricket " />  
  
<owl:Thing rdf:about="#Cricket">  
    <rdf:type rdf:resource="#Sport">  
</owl:Thing>
```

Figure 3.4: A simple entity defining Cricket.

3.1.4 Entities

Entities are simple instances that represent an individual member of a class. For example, Cricket is a member of the Sport class, which itself is a sub-class of Entertainment class. Hence, Cricket is also a member of the class Entertainment. *rdf:type* is the relation that establishes the link between the Cricket and Sport entities. This relation, in terms of Semantic Web, is defined as a property. Figure 3.4 demonstrates a simple entity.

3.1.5 Properties

In Semantic Web, property defines the one-to-one relation between an entity and the object that defines the entity [1]. Properties are mainly of two types:

Datatype properties: Relations defined by a generic String type object or RDF literals.

Object properties: Relations established between instances of two classes.

3.1.6 Relationships

While property may be the technical term, the theoretical demonstration is defined by relationships. Relationships define the link between the instances of two classes. As mentioned in section 3.1.4, *rdf:type* defines a specific type of relation. Where *rdf:type* is actually the property, it defines the relation 'IS-A' between the entity and the object. Figure 3.4 states that "Cricket is a Sport".

3.1.7 Ontology Mapping

Ontology mapping has been an integral part of Semantic Web. It basically states the inter-linking between heterogeneous data sources through some common concepts [34]. Previously, as Semantic Web used multiple ontologies for structuring the knowledge base across the web, ontology mapping was restricted to mapping between the ontologies themselves [35]. However, with the introduction of various features, that includes multiple language integration, the concept of one single ontology became very popular [6]. With the introduction of one single ontology, the ontology mapping got the new perspective of establishing relations among data sources. The data source mapping in DBpedia is quite a bit popular [5], as we will see in Section 3.1.11. It literally is defined as the relation that holds similar concepts for multiple entities. The example in Section 2.7 can be brought up in this case. The property “origen” in Spanish DBpedia and the property “origin” in English DBpedia for the entity “Earth” holds the same meaning, but in two different languages. Hence, the process of establishing a relation among these two properties can be termed ontology mapping.

3.1.8 Multilingual Ontology

As specified in Section 3.1.7, the concept of one single ontology has one of the main focus of it being multilingual. A single ontology that is language independent, has the capability to integrate new languages, adaptable to addition of new knowledge in the existing language domain is simply termed as a multilingual ontology [4]. As mentioned in [7], adaptation and addition of new knowledge bases should not result to any major changes in the multilingual ontology. Otherwise, the multilingual ontology loses its significance. Hence, the hypothesis was brought into reality by mapping the multilingual ontology with dictionaries and retrieving data from them [7, 36]. This resulted in integration of numerous languages without requiring any significant change in the ontology itself.

3.1.9 Resource Description Framework

Resource Description Framework or RDF is the standard framework, established mainly for the purpose of providing a basic model and foundation for creating and processing metadata [1]. It is recommended by W3C [26] and is most popular in defining ontology metadata. It describes

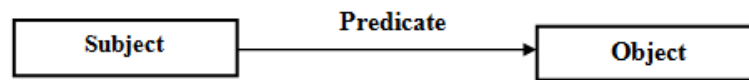


Figure 3.5: Structure of a triple format.

resources ignoring assumptions for a particular domain and hence, can describe any knowledge with the domains specified. The final model of RDF is completely machine-readable and in short, can provide a view to the machines as the traditional web provides to humans. A small piece of information in an RDF file is represented by a triple, formed by subject-property-object [1]. In case of an ontology or a dictionary, the entity itself acts as the subject. The object is an elaborate description, that can be assigned to any particular entity through a property. Each entity, property or an object can be of a URI form, which itself is defined further by adding more relations. The object can again be of literal type, where it directly provides some knowledge in String format. Figure 3.5 defines a simple triple format.

3.1.10 Wikipedia

Wikipedia a free multilingual encyclopedia, that is openly editable and is widely popular as an established data source. Human friendly user interface and loads of authentic references make Wikipedia one of the most dependable websites regarding knowledge extraction. With its inauguration in 2001, Wikipedia has integrated numerous languages and has evolved over time to accommodate more and more information. With about 290 Wikipedia encyclopedia integration, English Wikipedia still remains the richest of its resources. With authentic mapping techniques, the English Wikipedia has already been prepared in its machine-readable format and acts as a reference for innumerable researches relevant to Semantic Web. Having high demand across the world, new schemes are being added gradually to ensure the authenticity of the data represented in Wikipedia [24].



Figure 3.6: The official logo of DBpedia.

3.1.11 DBpedia

Wikipedia is the source of information for human eyes. When it comes to a machine, Wikipedia itself does not hold much significance, even though it is a vast repository of diverse knowledge. A machine-readable version of Wikipedia was established through the project of DBpedia in 2007 [37]. DBpedia simply puts forward the knowledge base in Wikipedia in a machine-readable format, and makes it accessible through the web [38]. With numerous languages being integrated in Wikipedia, and DBpedia being a direct extraction from Wikipedia, DBpedia became a highly established, large-scale knowledge base, which supports diverse knowledge in multiple languages [5]. As DBpedia can be accessed through the web, it provides support to systems developed in any programming language or using any framework. Thus, it is stable knowledge source that supports cross-domain knowledge extraction [17]. With so many pros, DBpedia is undoubtedly an asset in the field of Semantic Web and is a significant resource in innumerable researches.

While DBpedia itself can be accessed through the address `https://wiki.dbpedia.org/`, knowledge extraction from DBpedia requires that the abbreviation of the corresponding language will be appended with the URL. For example, the French and Spanish DBpedia can be accessed through `http://fr.dbpedia.org/` and `http://es.dbpedia.org/` respectively. However, as English is the base language for reference, the English DBpedia can be accessed directly through `http://dbpedia.org/`. It is possible to add non-existent DBpedias in future as well, like the Bengali DBpedia, through the URL `http://bn.dbpedia.org/`.

```

- <rdf:RDF>
- <rdf:Description rdf:about="http://dbpedia.org/resource/Lapis_lazuli">
  <ns7:hypernym rdf:resource="http://dbpedia.org/resource/Stone"/>
</rdf:Description>
- <rdf:Description rdf:about="http://dbpedia.org/resource/Main_Street_Bridge_(Rochester,_New_York)">
  <ns7:hypernym rdf:resource="http://dbpedia.org/resource/Stone"/>
</rdf:Description>
- <rdf:Description rdf:about="http://dbpedia.org/resource/Pennsylvania_Railroad_Bridge_over_Shavers_Creek">
  <ns7:hypernym rdf:resource="http://dbpedia.org/resource/Stone"/>
</rdf:Description>
- <rdf:Description rdf:about="http://dbpedia.org/resource/Tongji_Bridge_(Jinhua)">
  <ns7:hypernym rdf:resource="http://dbpedia.org/resource/Stone"/>
</rdf:Description>
- <rdf:Description rdf:about="http://dbpedia.org/resource/Tongji_Bridge_(Yuyao)">
  <ns7:hypernym rdf:resource="http://dbpedia.org/resource/Stone"/>
</rdf:Description>
- <rdf:Description rdf:about="http://dbpedia.org/resource/Ballingham_railway_station">
  <ns7:hypernym rdf:resource="http://dbpedia.org/resource/Stone"/>
</rdf:Description>
- <rdf:Description rdf:about="http://dbpedia.org/resource/Blaisdon_Halt_railway_station">
  <ns7:hypernym rdf:resource="http://dbpedia.org/resource/Stone"/>
</rdf:Description>
- <rdf:Description rdf:about="http://dbpedia.org/resource/Faringdon_railway_station">
  <ns7:hypernym rdf:resource="http://dbpedia.org/resource/Stone"/>
</rdf:Description>
- <rdf:Description rdf:about="http://dbpedia.org/resource/Glamis_Manse_Stone">
  <ns7:hypernym rdf:resource="http://dbpedia.org/resource/Stone"/>
</rdf:Description>
- <rdf:Description rdf:about="http://dbpedia.org/resource/Knowesgate_railway_station">
  <ns7:hypernym rdf:resource="http://dbpedia.org/resource/Stone"/>
</rdf:Description>
- <rdf:Description rdf:about="http://dbpedia.org/resource/Middleton_North_railway_station">
  <ns7:hypernym rdf:resource="http://dbpedia.org/resource/Stone"/>
</rdf:Description>
- <rdf:Description rdf:about="http://dbpedia.org/resource/Plashetts_railway_station">
  <ns7:hypernym rdf:resource="http://dbpedia.org/resource/Stone"/>
</rdf:Description>

```

Figure 3.7: A sample Data Set.

3.1.12 Date Set

The knowledge base extracted from DBpedia in a machine-reachable format are termed as Data Set. The term Data Set itself holds different significance when used in different context. As we are working on knowledge extraction from DBpedia, the data obtained directly from DBpedia are coined the term Data Set. Figure 3.7 demonstrates a sample Data Set of the entity “Stone” from the English DBpedia in RDF format.

3.2 Main Challenges in Focus

With the evolution of Semantic Web, a lot of researchers have been working on data extraction, ontology optimization and the multilingual ontology itself. However, the source data set optimization remains a branch that lacks enough research. While working on source data set

optimization, we faced quite a few issues, some which are listed below:

Non-existent DBpedia: Initially, we conducted our research based on the comparison between English data set, which still remains the richest among its kind, with Bengali data set. When we moved forward with our research, we found that, even though Bengali Wikipedia exists, Bengali DBpedia still lacks existence. Hence, we had to change our process by shifting the focus to French DBpedia instead. The lack of existence of DBpedia in majority of the languages creates a major drawback regarding the efficiency of the entire system. An ontology established through a scalable framework would have provided the facility to integrate languages in an easy way and thus, there would have been so many more DBpedias to select from.

Data Redundancy: While working on our research, we found that a lot of the properties in the data sets are redundant. Even though such properties hold the same meaning in different languages, the lack of mapping between them resulted in their co-existence as completely separate entities. The example in Section 3.1.11, that of “origen” in Spanish and “origin” in English is a direct proof of this issue. The framework for the ontology, if provided an easy mean to adapt new languages, would have taken care of such redundancies. When we enhanced the mapping system and applied multiple semantic rules, we found that the redundancy could be decreased to a significant level.

Lack of Mapping: Even though the entire data set of English Wikipedia is mapped with English DBpedia, none of the other languages are mapped up to such extent. As a result, the vast of knowledge in Wikipedia cannot be fully consumed by DBpedia. The mapping technique is one of the key features of the ontology framework. When new languages are adapted, the mapping technique should itself ensure the quality of the data sets. Without a standard set of rules applied through the mapping technique, the adaptation of new features faces a major barrier and thus, creates a significant drawback in the process.

Language Integration: Wikipedia is already supporting many languages. However, the knowledge base in DBpedia is poorer when compared to that. Due to lack of mapping, many languages does not have their corresponding knowledge repository in DBpedia. Non-existence Bengali DBpedia is a direct example of this issue. Language integration is

one of the key features that will contribute to the augmentation of DBpedia's knowledge base. After all, there will be no significant enhancement if there is no option to make the ontology capable of adding new languages.

Scalability: Due to data redundancy and lack of mapping, the current system is not scalable. Considering the English and Spanish DBpedia, the redundant data does provide a knowledge repository for Spanish language, but the vast amount of knowledge in English repository cannot support the Spanish DBpedia. Hence, although Spanish DBpedia exists, its knowledge base is significantly lowered and the scalability feature is compromised. Like mentioned in the previous point, until and unless the adaptation to new features is made easy, there will be no significance to make the world of web machine-readable. Otherwise, its only the same knowledge base that will remain understandable by artificially intelligent systems.

Efficiency: The lack of mapping results in the fact that the knowledge fetched for any language other than English is not the exact amount of data that is contained in the entire knowledge repository of DBpedia. As the current system uses separate dictionaries for each individual language, the multilingual property is maintained. However, a good amount of knowledge goes to waste when only a single dictionary is considered for a particular entity. We have found in some cases that the amount of knowledge in the second language is greater than that of English itself. If we can merge the data sets together, even English knowledge repository will get richer. Logically, if a single data source can provide a deck load of data in comparison to the older data source, the efficiency of data retrieval will improve, as well as the amount of data content against each entity. Thus, the efficiency of the data set will make the entire knowledge repository richer. This acted as one of our major motivations for this research.

The above mentioned issues can be significantly overcome through the efficient usage of the system we established through our research. By submerging the source data set of two languages, we are ensuring that both the languages can fetch same amount of data for any particular entity. Thus, removing almost all of the issues we mentioned above. A further possibility of enriching can be considered when we will be able to merge the combined data set with the data sets of more languages. This process will repeatedly augment the knowledge

domain of all the languages that are merged. The ultimate result will enrich DBpedia and simultaneously, augment the knowledge domain of Wikipedia.

Chapter 4

Methodology

4.1 Proposed Methodology

Our proposed methodology and its subsequent implementation are mainly divided into two separate hypotheses. Each hypothesis has its own contribution in establishing the entire mechanism. The first hypothesis denotes the framework designated for a multilingual ontology [7]. With its help, the designated data sets are taken into further consideration. Here comes the second hypothesis, which augments the entire domain of Wikipedia and DBpedia [12].

The two steps are termed as:

- Framework for Wikipedia entry augmentation.
- Submerging multilingual entities.

The hypotheses will be elaborated in the following sections.

4.2 Framework for Wikipedia Entry Augmentation

The data set of Wikipedia is quite sufficient for some major languages. English is the most enriched language till date [39]. Since DBpedia is based on the data set of Wikipedia [5], it can be suggested that English is the richest data set for a single ontology.

We put forth here the concept to establish a framework to augment the Wikipedia data set. According to the statistics previously presented in Table 2.1 and Table 2.2, Bengali has been

found to be one of the languages yet to be brought under the domain of a single ontology. We propose here the main concepts required for the augmentation.

To augment the Wikipedia data set, as mentioned in Section 1.1, the initial step is to implement a single ontology, which can use dictionaries for each language as sources, and link up all the data on its own. Hence, a multilingual ontology is the first step. Researchers have developed the basic framework for the multilingual ontology [40]. The next step is to enrich the data set of the ontology. In such a case, a mapping technique is implemented, which links up all the words or entities, from different languages, with the same meaning [41, 42]. The ontology then creates a universal identifier for each particular concept. The final step is to retrieve the information for using in any web content. An addition to this step is the rendering of web content, which allows the user to adopt a simpler way to link up the content with the ontology.

4.2.1 Multilingual Ontology

The concept of multilingual ontology follows that, there should be one, highly enriched ontology. The ontology should have the capability to utilize infinite number of dictionaries. Thus, these dictionaries are going to enable the ontology to cope up with multiple languages [6]. At the moment, we are assuming that there is one dictionary for each language. The multilingual ontology acts as a formal and explicit specification of a concept, which can be shared by all existing and upcoming dictionaries [43]. It consists of a set of distinct concepts, having inter-relations, denoted by a set of relations [6].

We will put a simple example here to further elucidate the above concepts. We have already used the example of *Rice* in Section 2.1. Considering *Rice* as an English word, the definition is derived as “*A swamp grass which is widely cultivated as a source of food*”¹. When we consider this in reference to the ontology, it has a property named *type*, with the object value as *grain* [44]. The idea of a concept is the class *rice*, which can be back traced to the parent class of *thing*². Again, another concept can be similarly stated as *wheat*, having the definition “*A cereal grain that yields a fine white flour*”³. This particular concept also has the property *type* with value *grain* [44]. Hence, the two concepts are connected with a single relation, i.e.

¹<https://en.oxforddictionaries.com/definition/rice>

²http://neon-toolkit.org/w/images/Doku_2.3.pdf

³<https://www.merriam-webster.com/dictionary/wheat>

grain, which itself is stated as a “term”. The implementation of multilingual ontology states that the concepts are stated by a unique identifier in the ontology itself, and the descriptions are fetched from each individual dictionary. The identifier for *Rice* is <http://dbpedia.org/resource/Rice>, and all its relevant information are obtained from the English data source in <http://dbpedia.org/data/Rice.rdf>. However, when the same object is accessed for a different language, consider Bengali for instance, its translated version “চাল” is linked up with <http://dbpedia.org/resource/Rice>. However, in this case, the relevant data is to be fetched from the Bengali dictionary or the Bengali data source, which is <http://bn.dbpedia.org/data/Rice.rdf>. As the Bengali DBpedia lacks existence, we can also consider the french derivation. The French translation being *riz*, the data source becomes <http://fr.dbpedia.org/data/Rice.rdf>.

Existence of such an ontology eases the opportunity to integrate newer languages. Thus, the integration of a language-based ontology augments the data set of Wikipedia.

4.2.2 Multilingual Ontology Mapping

As previously mentioned in Section 2.1 of this thesis, the mapping of multilingual ontology acts as the most significant part for the augmentation of Wikipedia. We need to consider the existence of a multilingual ontology. A process that requires frequent changes to the multilingual ontology will never be an efficient way. Thus, the requirement for an efficient mapping technique is of utmost significance.

For our research, we are proposing a technique, based on [4], that takes an established dictionary as source, given that the dictionary is already integrated with the multilingual ontology, and map a new dictionary with it. This process extends the universal identifiers that are linking the ontology with the existing vocabulary, to the new dictionary, linking up its vocabulary. Techniques described in Section 2.2.1 are adapted in our context to achieve the mapping technique.

To augment the entries, first we require an ontology of the new language. Considering it as *new ontology*, we need it to be uploaded to a *medium*, which is actually a web interface. This medium simply acts as a window to provide the new ontology to the existing system. Upon receiving the *new ontology*, a translation mechanism is conducted, to generate an ontology

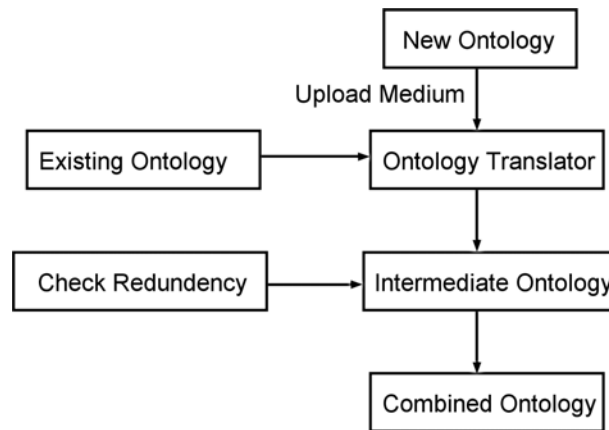


Figure 4.1: Generation of combined ontology.

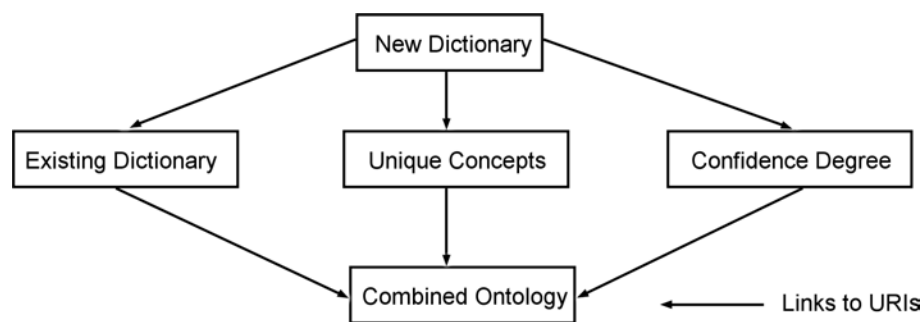


Figure 4.2: Linking dictionary to ontology.

similar to the *existing ontology*. From this stage, a checking for redundancy is conducted. Any concept already existing in the *existing ontology* is discarded, and new concepts are added. The final, *combined ontology* is basically a combination of the *existing ontology* updated by addition of some new concepts [6]. Its generation is sequentially demonstrated in Figure 4.1.

At this stage, we establish mapping between the two dictionaries. For this, we require translation from the *new dictionary* to the *existing dictionary*. Let us consider that we have Bengali dictionary as the *new dictionary* and English dictionary as the *existing dictionary*. After translation, we map the direct translations with the universal resource identifier that connected the corresponding English vocabulary. This ensures that no redundancy exists. For the portion that does not come to much effect for direct translation, the new concepts added to the *existing ontology* will cover them up. The sequence is shown in Figure 4.2.

To map the vocabularies in the dictionary, which are not providing any specific result for direct translation or may not be suitable for linking with the new concepts, we require some measure to find out the closest match. For such cases, the mapping proposed in [4] can be used,

which provides a *confidence degree* [4]. The best match, judged by the *confidence degree* is to be linked up with the universal resource identifier.

4.2.3 Information Retrieval

We can retrieve information from the mapped ontology by constructing SPARQL queries directly to the SPARQL ENDPOINT⁴ or by making textual queries which can be converted into SPARQL queries later. Just as the multilingual ontology comprises of a set of mapped ontologies from various languages, retrieval of information also requires to be language independent. The mapping technique, proposed in Section 4.2.2, for augmenting multilingual ontology can also be adapted in this case.

To make the retrieval system language independent, the textual queries require to be translated into an existing language from which the conversion to SPARQL query becomes easier. Considering the similar example as used Section 4.2, English language is used as the existing one, and Bengali language as the local one to demonstrate the implementation. Whenever a textual query is made in Bengali, a tag needs to be attached to state the language. Consider the tag for Bengali to be <BN>. Such tags help the system to identify the language.

At first, the Bengali textual query is translated into English textual query using a regular language translator. Then the English textual query is tokenized into entities. Open Calais⁵, Twine⁶, Zemanta⁷ etc. are some of the products, which are built on Named-Entity Recognition (NER). These provide APIs which can be used to extract entities. These entities are serving as the unit parts of the query. Then the SPARQL query is formed using these entities.

An example can be illustrated for better understanding of the concept. Let, a query is made in Bengali - “১৯৩০ সালের পরে জন্ম এমন বাংলাদেশী অভিনেতাদের নাম?” (“list of Bangladeshi actors born after 1930”). The expected result will be the list of URIs of the Bangladeshi actors who were born after 1930. To process this query we first translate it into English. The translated query is “After 1930, the name of the actors who were born in Bangladesh?”. The extracted entities are *birthPlace: Bangladesh, occupation: Actor, Name, birthDate > 1930* etc. So, a SPARQL query is generated from these entities. The generated SPARQL query is shown in Listing 4.1.

⁴<http://dbpedia.org/sparql>

⁵<http://www.opencalais.com/>

⁶[https://en.wikipedia.org/wiki/Twine_\(website\)](https://en.wikipedia.org/wiki/Twine_(website))

⁷<http://www.zemanta.com/>

```

PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX db-ont: <http://dbpedia.org/ontology/>
SELECT DISTINCT ?person ?name ?birth
WHERE {
  ?person db-ont:birthPlace <http://dbpedia.org/resource/Bangladesh> .
  ?person db-ont:occupation <http://dbpedia.org/resource/Actor> .
  ?person foaf:name ?name .
  ?person db-ont:birthDate ?birth
FILTER (?birth > "1930-01-01"^^xsd:date) .
}

```

Listing 4.1: Generated SPARQL query.

Since the English and Bengali vocabularies are already integrated to the multilingual ontology, the universal identifier returned by the translated query is thus, the same for the object the Bengali query was initially formed for. If we only want the results written in Bengali language, then, we need to add a language filter in the SPARQL query.

4.2.4 Data Sets of the Ontology

As stated in Section 4.2.3, SPARQL queries can be applied to fetch the knowledge base from DBpedia. The standard format for consideration with regard to this knowledge base is the triple format of subject-property-object [1]. Hence, when we state that the data sets obtained from DBpedia, we specify the subject-property-object triples against a particular entity. We will find, as we move forward with this thesis, that the extracted data can be processed further to enhance the domain of Wikipedia. For this particular purpose, we will have to extract the knowledge base and generate data sets that are compatible with the system proposed in our thesis.

4.3 Submerging Multilingual Entities

The framework suggested in the previous section [7] states that the concepts defined in each individual ontology should be mapped between one another. Based on the hypothesis stated in [12], it is safe to assume that the repetition of properties causes the repetition of URIs in different DBpedia resources. Hence, the mapping process will ensure that the repeated properties are considered exactly once.

4.3.1 Language Selection

In order to demonstrate the working procedure of submerging multilingual entities, we are adapting examples from two languages. Table 2.4 evidenced a clear difference between the mapped entities of English and French languages. Thus, we are using these two languages as our frame of reference. Based on our experiments, we found that it is not always necessary that the number of properties describing a single entity in English language will be more than those describing its corresponding entity in French language. In most cases, it was found that the number of properties does not match and there are always some unique properties in both English and French languages. In order to get a clearer idea, we will move on with the work flow by considering a single entity from French language and its English derivation.

Let us consider the object “Lune” from French DBpedia. As already stated, there are separate RDF documents hosting the properties for “Lune”⁸ and its English derivation “Moon”⁹. Our objective is to merge the set of properties from both the documents.

4.3.2 Translation and Mapping With DBpedia

Initially, the French word “Lune” is provided to the system through an interface. Upon receiving the input word, the system runs it through a translator. The translator provides the English derivation of the input word, i.e. “Moon”. The next step is to generate the subsequent DBpedia links for each RDF document. The system generates two well structured links or web addresses [1], one for each of English and French DBpedia. The structure of the link defines that the link needs to start with the corresponding DBpedia address, <http://dbpedia.org/> for English and <http://fr.dbpedia.org/> for French. Since the RDF documents act as the data set for DBpedia, thus, the link is then followed by the keyword `data`. Finally, the link is ended by adding the English word for both cases and `.rdf` as the extension to denote RDF file. The final links for “Lune” and “Moon” stands as <http://fr.dbpedia.org/data/Moon.rdf> and <http://dbpedia.org/data/Moon.rdf> respectively.

⁸Moon RDF file from French DBpedia, <http://fr.dbpedia.org/data/Moon.rdf>, accessed on 17-April-2018

⁹Moon RDF file from English DBpedia, <http://dbpedia.org/data/Moon.rdf>, accessed on 17-April-2018

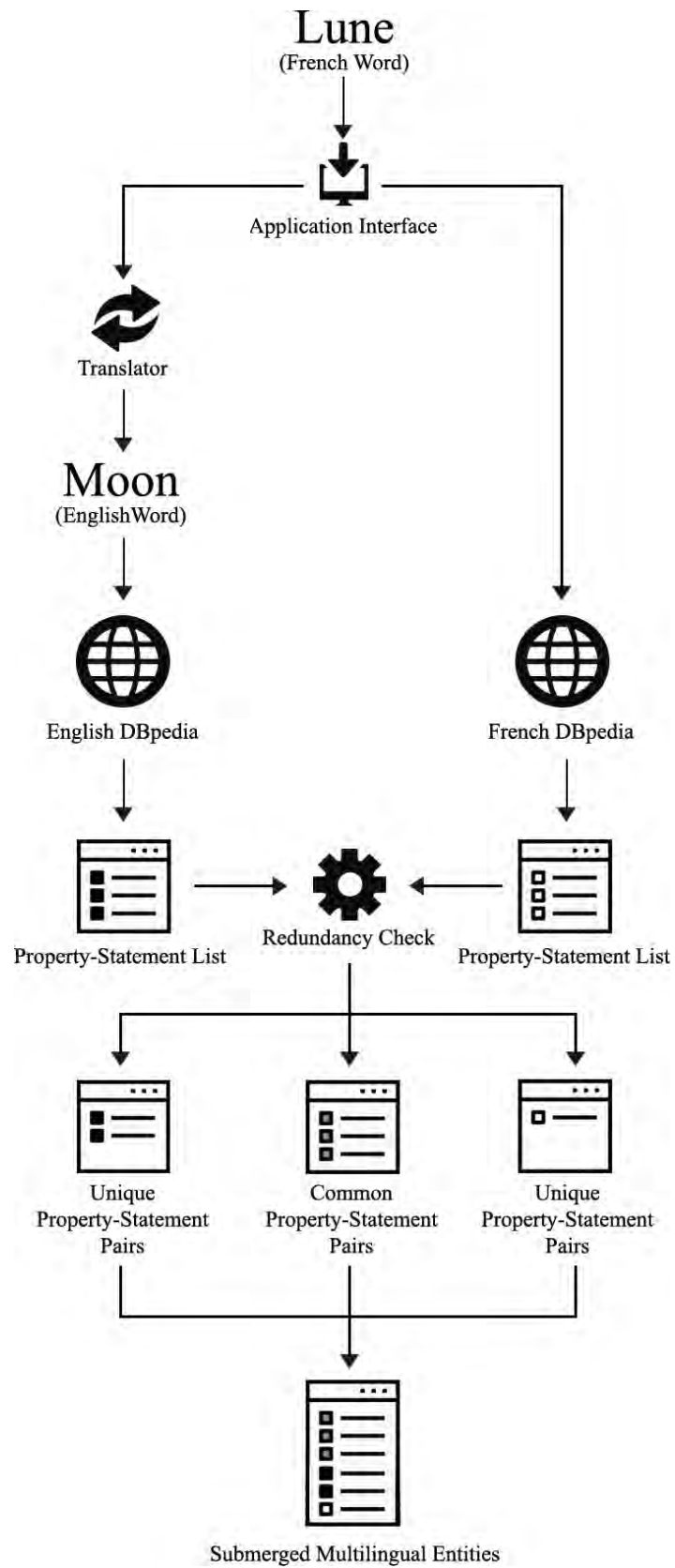


Figure 4.3: The process of submerging multilingual entities.

4.3.3 Extracting Knowledge Base

At this stage, the system generates SPARQL queries to fetch all the properties listed in each document. The queries are generated in a way, such that the result set contains the property-statement pairs for each entity. This is done so because the main reference of comparison between the properties of Lune and Moon is considered as the combination of property and statement. The property and the statement bear the same significance as the property and object part in a triple format of subject-property-object [1].

4.3.4 Preparing Data Sets

As we are considering two separate versions of DBpedia, two separate sets are created, one for the property-statement pairs in the English RDF file and another for that in French RDF file. The statement portion of the property-statement pair has the value in one of two types. These two types make up two of the following steps adapted to prepare the data for further processing.

- The first type is a literal value, where the value is simply raw text data. The other type is URI, where the value is denoted by a unique identifier [1].
- If the fetched statement value is of literal type, its lexical form is to be recorded in the data set. Such as, the statement for the property “name” in English RDF file is “Moon”, which in this case, is a literal value. Hence, the system records its lexical form, i.e., the string “Moon”.
- If the value is of URI type, the system first tries to extract its local name. A simple example is, the value of the property “type” in the English RDF file is “<http://dbpedia.org/class/yago/CelestialBody109239740>”, but a local name can be derived from this URI. Thus, the local name of the URI, “CelestialBody109239740” is recorded as the string value of the statement in the corresponding property-statement pair.
- If, however, a local name is not extractable, then the system directly records the URI. The statement of the property “wasDerivedFrom” in the English RDF file is “<http://en.wikipedia.org/wiki/Moon?oldid=707145816>”. Since a local name cannot be derived from this URI, the system records the entire URI as the statement.

- An additional comparison criterion is taken into consideration, by fetching the language tag (if available), to denote the language of the statement. Such as, the language tag of the statement “Moon” for the property “name” states English in a two character abbreviation, “en”.

Once the data set from the English RDF file is prepared, the data set from the French RDF file is extracted. This process, again, is conducted through the following steps:

- Like that of the English data set, if the value of the statement is a literal, the lexical form is recorded.
- However, the value is further checked in case if it is a decimal number. In such a case, the language tag is automatically updated to “en” for English.
- Similar to the case of English data set, the language tag (if available) is also fetched for each property-statement pair.
- Now, if the value of the statement is a URI, the system tries to extract the local name first.
- In case if the local name is available but the language tag is empty, the system conducts another checking to detect the possible language of the value. If the language with maximum probability is English, the language tag is updated to “en”. Otherwise, the system checks if one of the probable language is French. If so, the system updates the language tag to “fr” to denote French.
- Lastly, if the value is a URI with no local name, and if the language tag is empty, the system checks if there is any mention of known French or English source, like `http://fr.dbpedia.org` for French and `http://en.wikipedia.org` for English. In such a situation, the language tag is updated with the corresponding value.

Once the data sets are created, the system then moves on to generate the sub-merged data set by passing through a comparison mechanism.

4.3.5 Set of Semantic Rules

Based on our research, it was found that all the instances of English language are mapped. Hence, the submerged data set is initially loaded with all the property-statement pairs, further

defined by language tags, fetched from the English RDF file. Now, each property-statement pair from the French data set is traversed in a linear fashion, and compared with all property-statement pairs of the English data set. For any French pair, there are three steps of consideration:

1. The first consideration is a direct comparison of all the three values in hand, viz. property, statement and language. If for any French pair, there exists an English pair with a perfect match of all three values, the French pair will be considered as redundant, unless it is further considered by any of the two following criteria.
2. If the value of the statement is of literal type, and if the language tag states “fr”, the value is translated to English. The translated value is then cross-checked with each statement from the English data set. In case of a perfect match, the property of the French pair is updated with the property of the matched English pair, while the statement remains in French. The language tag also remains as “fr”, denoting that the French pair is actually a language variation of the English pair. The updated French property-statement pair is cross-checked with the entire English data set, and is added to the sub-merged data set only if no perfect match is found.
3. Finally, if the value of the statement is a local name of a URI and with the language tag of “fr”, the system cross-checks the value from the French pair with the value of the statement in every English pair. In case of a match, the property of the French pair is updated by replacing it with the property value of the English pair. The statement and language tag remains unchanged for the French pair, and thus, acts as a language derivation of the English pair. The updated French pair is again cross-checked with every pair of the English data set, and is added only if no match is found.

If a French pair receives a true value from the first criterion and is not considered by the other two criteria, then it is confirmed as a redundancy, and is not added to the final sub-merged data set. All other pairs are sequentially added and thus, the sub-merged data set consists of the English pairs, followed by the non-redundant French pairs.

4.3.6 Complexity Assessment

As an example, let us consider the property-statement pair `label-Lune`, with language tag `fr`. This particular property, after being fetched from French RDF document, is compared to each property of English RDF document. Once an identical property is found in English RDF document, i.e. `label`, the statement or value corresponding to the property is compared. If the statement describing this property is indeed `Lune`, and the language tags for both the pairs are same, i.e. `fr` the property-statement pair may be taken as redundant. Since the statement is of literal type and the language tag states `fr`, the pair is considered in the second stage of checking. The value of the statement is translated to English and `Moon` is received. A checking through the English data set for statement value of `Moon` is conducted and a match of `label-Moon` is found. Hence, the property for `label-Lune` is updated to `label`, though it remains the same in this case. Now a final checking is made with `label-Lune` with language tag of "fr". If such a pair exists in the English data set, the pair is taken as redundant. Otherwise, it is considered worthy of addition to the submerged data set. In the final checking phase, the pair `label-Lune` is not considered as the statement is not a local name of a URI. During our experimentation, the property-statement pair `label-Lune` was found to be redundant in the first phase and also in the final checking of second phase, and hence, removed from the French data set. The property-statement pair would persist if it was not redundant. Since all the nested loops in the comparison mechanism takes place at a maximum depth of 2, it will be conducted in the order $O(n^2)$ for a single entity in 2 languages at a time, where n is considered the universal case for the number of iterations for a single level of one loop. However, when we consider n languages, the complexity will be $n - 1$ times of that for 2 languages. Hence, it stands as $O(n - 1)(n^2)$. Now, if we consider m entities in the richest of n languages, the complexity stands of order $O(m(n - 1)(n^2))$. This may not be much significant when we consider that the operation needs to be conducted once for each entity, that is m number of times. Moreover, the text matching is a linear mapping between the objects or statements of the triples. This ensures that the text matching is conducted with the minimum amount of complexity.

The final point of consideration is the scenario when new entities are added to the existing data sets, after implementing the proposed submerging technique. Irrespective of all languages, the data can be inserted into the current data set of Wikipedia through any existing method. To ensure that the data is mapped with DBpedia, however, the data insertion processes are

restricted to only a handful. The processes elaborated in [45] are quite standard for ensuring the map between Wikipedia and DBpedia entities. The English entities can be added without any further concern. For any other language, once the data is inserted, the submerging system needs to be executed to ensure its mapping with English data set. As only one entity will be considered for two languages, the complexity in this case will also be $O(n^2)$.

4.3.7 Finalizing New Data Set

The comparison operation removes all the properties from the French data set that are redundant, keeping only the unique ones. A new data set is formed with the properties from the English data set, followed by the unique properties from the French data set. This final data set is the submerged property list. These properties can all be referenced through the derivation of a single identifier and a single ontology will be able to maintain all the definitions.

Figure 4.3 demonstrates a step-by-step flow of the entire process of submerging multilingual entities.

4.4 Summary

In short, our hypotheses can be listed down for the final methodology in the following way:

- The current multilingual ontology is modified to include multiple data sets.
- The extraction techniques are to be implemented to ensure that the knowledge base is properly mapped and the extracting technique agrees with the mapping.
- One rich and mapped language and another language, which is comparatively new, are to be considered and one specific word is selected and translated for adaptation in both languages.
- The words are mapped with DBpedia following specific rules to form URLs.
- The knowledge base for both the words are extracted using standard extraction techniques.

- The data sets are prepared from the knowledge bases so that they are compatible for further processing.
- The set of semantic rules are implemented on the processed data sets.
- The final and augmented data set is created by the resulting knowledge base obtained after implementing the set of semantic rules.

Chapter 5

Experiments

The proposed system includes a good amount of dependency on communication between multiple systems and platforms over the internet. Hence, we have chosen Java as the development language as the interfacing capability of Java¹ is unparalleled. The initial user interface is developed using JavaFX², providing a simple desktop portal to provide the system with the specific entity. The translator API provided by Google Translate³ has been used as the translator for the proposed system.

5.1 Competency of the System

The framework for multilingual ontology proposed in [7] will provide the best result on application of the submerging mechanism proposed in [12]. However, it should also be mentioned that the submerging mechanism yields high quality results whenever applied to any framework concerned with multilingual ontology. As the results will demonstrate for our application, the submerging mechanism guarantees an improvement in the quality of the result, and its subsequent augmentation of DBpedia.

¹<https://docs.oracle.com/javase/8/docs/>

²<http://www.oracle.com/technetwork/java/javafx/overview/index.html>

³<https://cloud.google.com/translate/docs/>

5.2 Selection of Language and Translation

Implementation of the system initially required an interface that took input of specific format from any language in consideration. In our case, the Java interface is able to take input in French language. Upon receiving the input, the system uses the translator mentioned in Chapter 5, to get the corresponding word in the language which is richest in respect to structured and mapped entities. As already mentioned in Section 2.7, the most convenient language for such a case is English. Hence, when the system receives the French word, it passes the word through an online translation tool (Google Translate API) and gets the English translation. All further steps use this translated version, as well as the original input.

A reason for the consideration of French language during implementation was that the entities or instances of French data set have majority covered by canonicalized data set with English instances [23]. Since there exists a specific structure of identifiers for French instances, it would be possible to generate the URI of the French instance based on the translation.

In the following sections we are going to demonstrate how the system fetches all the properties from the RDF file of English DBpedia and then, the properties from that of French DBpedia, both following the principles mentioned in Section 4.3.3 and Section 4.3.4.

5.3 Populating Data Sets

Our experimentation showed that there may be a few properties which are already mapped to the properties of English data set. Such properties are repeated in the set of properties fetched from the French DBpedia. Our experimentation brought forward this observation.

Initially, the system generates the link corresponding to English DBpedia for the particular entity in concern. SPARQL queries are generated in accordance to the generated link. Execution of the SPARQL queries makes a quick search over the internet for the link in DBpedia. Once found, the thread corresponding to the execution of the query brings back the data for the entity. The data is strictly formatted in subject-property-statement form. Additional field of language tags are also fetched for each entry.

Once the English data set is populated, the system follows the similar manner to generate the link for the French DBpedia. Corresponding SPARQL queries are formed and executed through

a separate thread. This thread again makes a quick search over the internet and returns with the data for the French entity. This data also maintains the format of subject-property-statement, along with the language tag.

5.4 Processing the Data Sets

As both the data sets are fetched, a set of logical methodologies are applied to them, as mentioned in Section 4.3.4. At first, the English data set is put into consideration and the system traverses through the entries one at a time. Once the application of logic is completed, the resulting data set of English language are separately distinguished on the basis of the type of statements. The literal and URI type statements are clearly designated in different manners. For the entries having a local name, the local name is considered as the statement. This process completes the first of the two steps in processing the data sets.

The French data set is put into consideration after that. Similar to that of English data set, each entry is traversed and the type of statement is designated. As a result, the system identifies all literal, local name and URI type statements. An additional mechanism is implemented in this case. For any literal or local name, the system checks for possible language matches. As we can only assume the language we are considering, the system assigns French language tags only, if the probability matches. For URI type statements, the system makes a quick parsing of the page redirected by the URI. If the content is HTML type, the system goes through the text and assigns the French language tag if it matches.

After the entire logic implementation is completed, we have two processed data sets that are eligible for the application of semantic rules.

5.5 Application of Semantic Rules

A major focus of this thesis was the generation of the semantic rules stated in Section 4.3.5. The application of this set of rules ensures that the final data set is free of redundancy and is unique in respect to each entry. As even a single repetition will result to an inefficient augmentation of data, the semantic rules have been created and tested to ensure the highest percentage of efficiency. Each one of them is implemented in its logical form using Java. As we have the

processed data sets ready at this stage, both the data sets are fed into the block implementing the rules.

Since English data set acts as the reference, the system checks each French property with each English property. Initially each obtained value against an entry is compared between the English and French data sets. If the value of property, statement and language tag all found to be redundant, the entry is put into further consideration with the next rule. Once the entry is discarded by all the rules stated in Section 4.3.5, the pair is taken as a repetition. All repeated combinations are removed from the French data set by the system.

5.6 Submerging Data Set

At this stage, the submerged data set is already loaded with all the pairs from the English data set. We also have another data set, derived from the French data set, which is a list of unique property-statement combination, with the total number less or equal to that when initially fetched. The system now applies a merging procedure and the previous submerged data set is appended with the data set derived from the French one.

This new data set is comprised of unique property-statement combination, all describing the particular entity under consideration. The total number of sub-merged property-statement combination will be greater or equal to the number of property-statement combination obtained from English data set. This submerged data set can now act as data source for both the DBpedia editions, independent of the language called from.

5.7 Preparing the Export File

Additionally, for a better presentation, the system developed by us alphabetically sorts the list of property-statement combination on the basis of the properties. Hence, the submerged data set is put into the sorting mechanism and we get a well-structured source data set.

Here, we implemented a Java library, named JDOM⁴. This library is capable of generating XML files. We followed the standard convention from DBpedia, and generated an export file with similar representation. Figure 5.1 demonstrates the parent RDF tag opening the document.

⁴<http://www.jdom.org/downloads/docs.html>

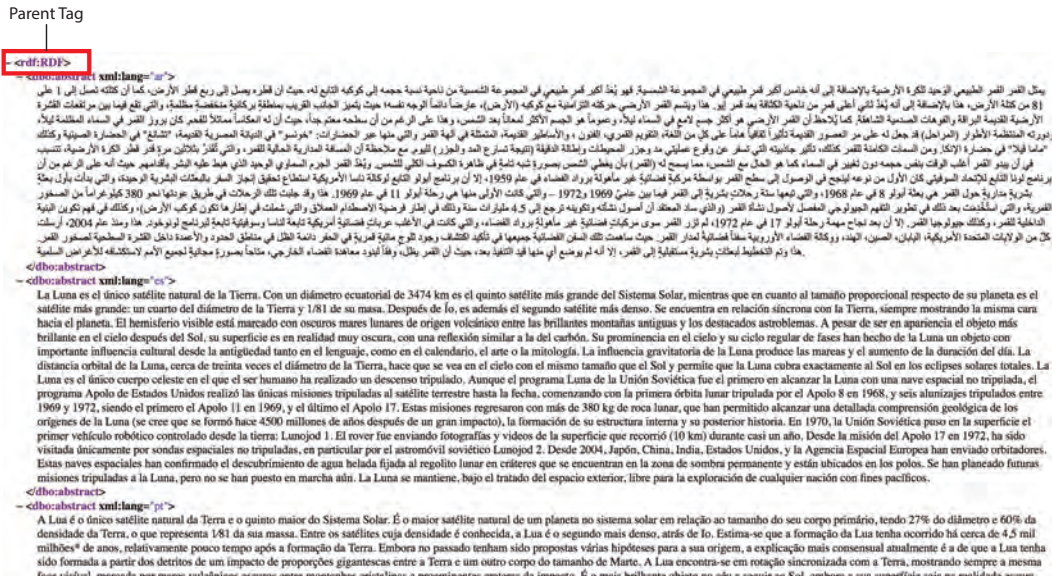


Figure 5.1: The parent tag as found in the generated XML file.

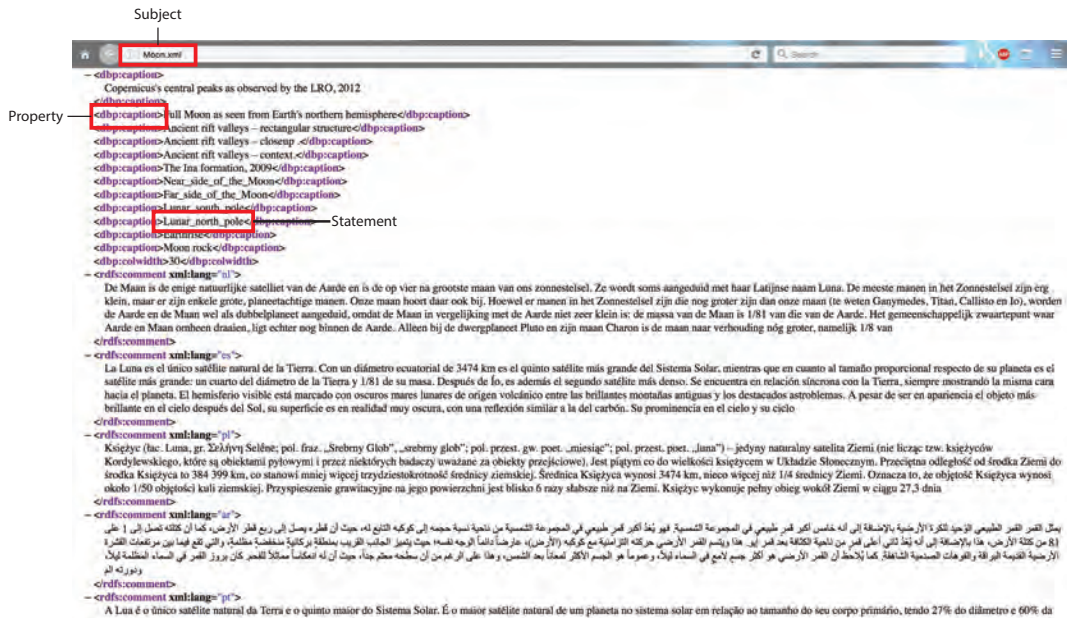


Figure 5.2: The subject-property-statement format represented in the exported XML file when opened in Mozilla Firefox.

This RDF tag is derived from the namespace “rdf”, which is the XML representation for <http://www.w3.org/1999/02/22-rdf-syntax-ns#>. Figure 5.2 shows how the triple format of subject-property-statement is represented in the file.

The exported XML file is completely suitable for uploading to any web domain and the data can be accessed in the same manner as that of DBpedia.

Table 5.1: Experimental results obtained by implementing the process of sub-merging multilingual entities in case of French language.

Input, fr	Translation, en	Properties, fr	Properties, en	Common Properties	Sub-merged Properties	Percentage Increase, fr	Percentage Increase, en
Lune	Moon	89	203	18	274	207	35
Soleil	Sun	196	201	21	376	92	87
Terre	Earth	50	164	8	206	312	26
Homme	Man	110	66	14	162	47	145
Poisson	Fish	112	115	7	220	96	91
Vent	Wind	50	84	5	129	158	54
Oxygene	Oxygen	92	94	6	180	96	91
Lumiere	Light	37	63	8	92	148	46
Eau	Water	44	86	3	127	189	48
Sourire	Smile	111	60	18	153	38	155
Bienvenue	Welcome	53	63	17	99	87	57
Je	I	123	97	7	213	73	120
Début	Start	45	57	17	85	89	49
Son	Sound	10	79	1	88	780	11
Maison	House	108	72	16	164	52	128

Table 5.2: Experimental results obtained by implementing the process of sub-merging multilingual entities in case of French language.

Input, fr	Translation, en	Properties, fr	Properties, en	Common Properties	Sub-merged Properties	Percentage Increase, fr	Percentage Increase, en
Mère	Mother	66	67	14	119	80	78
Père	Father	27	81	2	106	292	31
Peine	Sentence	27	16	3	40	48	150
Histoire	Story	0	42	0	42	new	new
Réponse	Answer	9	33	1	41	356	24
Montagne	Mountain	52	69	8	113	117	64
Couleur	Color	57	83	6	134	135	63
Exemple	Example	30	22	2	50	67	159
Riz	Rice	150	99	12	237	58	139
Papier	Paper	39	89	7	121	210	36
Rivière	River	9	70	2	77	756	10
Oiseau	Bird	121	137	13	245	102	79
Feu	Fire	73	87	9	151	106	74
Sud	South	0	63	0	63	new	new
Bleu	Blue	137	85	4	218	59	156

5.8 Experimental Result

The process described in Chapter 5 is implemented in case of French language. Table 5.1 and Table 5.2 shows the experimental results obtained by using multiple French words, three of which are Lune, Soleil and Terre, when translated to English, means Moon, Sun and Earth. Based on analysis of the number of original properties in the candidate languages and those in the submerged set, the percentage difference is also summarized in the Table 5.1 and Table 5.2.

While just considering one single case, Lune, it is clearly observable that the two RDF documents has different set of properties. As already stated, the English data set is quite rich for Wikipedia and hence, DBpedia. However, the sum of both the set of properties, if all were unique for each set, would have been 292. The experiment shows that there are specifically 18 properties that are completely similar. Thus, the system successfully removed the redundancies. All the 274 properties listed by the execution can be termed as unique.

Furthermore, if the 274 merged properties are used through a single source for the knowledge base, the Wikipedia page which was providing details based on the 203 properties from the English data set will have an additional 71 properties to fetch data from. Thus, giving it an approximate of 35% increase in data source. Similarly the French Wikipedia page will have an approximate increase of 207%. Merging of only two languages can simply result to this huge increase in knowledge amount, for just one single word. The experiment showed positive increase for all tested cases, only a handful of which are summarized in Table 5.1 and Table 5.2. The experiment proves that there is a 100% possibility of the submerged data set being richer than any individual data set. Hence, using a single set as the source for all unique URIs, the redundancy can be removed and the entire set can be made machine-readable.

In order to justify the authenticity of the generated result, the generated XML files were run through multiple validation engines available over the internet⁵⁶⁷. The validation engines were ensured to be genuine and their authenticity was judged by the amount of knowledge base they hosted which is relevant to XML itself and its usage in other researches [46]. Initially the data sets fetched from DBpedia were passed to the validation engines, followed by the XML output file generated by the system proposed in this thesis. For all cases, the XML files were stated to

⁵<https://www.xmlvalidation.com/>

⁶https://www.w3schools.com/xml/xml_validator.asp

⁷https://www.truugo.com/xml_validator/

Point of Consideration	Attainment
Number of overall language consideration	Infinite
Limit on number of data sources for consideration	None
Consideration of possible data sets	Both existing and non-existing
Percentage of cases improved	100%
Cases with redundant data	0

Table 5.3: Proposed performance metrics for executed system.

be authentic. Therefore, considering that the generated XML files can be parsed in a manner similar to the data extraction technique of DBpedia [47], the comparison metrics have been established as shown in Table 5.3. In a nutshell, it can be stated that, based on the authenticity of the data set of DBpedia, the metrics proposed in Table 5.3 will hold true for all data sets generated by the proposed system in this thesis.

Chapter 6

Conclusion and Future Work

6.1 Compendium of Attainments

In this thesis, we have introduced a framework for multilingual ontology and a submerging technique that can augment the knowledge base of DBpedia, and with extension, Wikipedia. A brief description of the notable features of our proposed methods is as follows:

- The framework for multilingual ontology proposes the capability of the ontology itself to adapt infinite number of languages.
- The multilingual ontology is capable of being mapped with dictionaries or data sources of infinite number of languages.
- The submerging technique is able to map with any data set of existing DBpedia. In case of non-existent DBpedia, providing the link to the empty DBpedia will suffice.
- Once applied, the submerging technique is able to fetch the existing data sets and generate a new data set that is guaranteed to be either equal to or richer than the richer data set among the two.
- Our proposed technique applies a set of well justified rules that ensures the data in the submerged data set provides highest efficiency.
- The set of rules removes the major issue in the current domain of DBpedia, which is the repetition of entities having the same meaning, but are from different languages.

- Apart from the main proposition, our system also prepares the data set in XML format, which can be easily adapted by the current system. We have also ensured that the knowledge in the final data set is well organized and sorted.

6.2 Integration of New Languages

The system developed by us was able to successfully merge and provide a data set that is either equal to or rich than the existing one. The merging process is compatible in case of two languages. Any future progress should put into consideration the objective of removing the constraint on number of languages. This simply means that the updated system must be able to submerge the entities from all languages that has a mapped definition in DBpedia [23].

6.3 Scope for Bengali DBpedia

Currently there are so many websites which are rendering their large amount of web contents using semantic web technology, for example, BBC [48], Best Buy etc. Without it, none can be greater than the sum of its own pieces of information.

From the detailed discussion on the framework and the source data set enhancement technique discussed in this thesis, it can be stated that an enhancement of Bengali DBpedia is quite possible. The ontology for Bengali language needs to be generated initially. It has to be mapped with the prevailing multilingual ontology. The submerging technique can then be applied with the combination of English and Bengali data sets and the Bengali DBpedia will get its initial knowledge base. After that, the queries can be generated in any language to get the Uniform Resource Identifier (URI) for each object. Using this ontology, website contents which are written in Bengali languages can be rendered automatically in future.

The significance of the implementation can be described using one simple example. Figure 6.1 shows only a small portion of the page representing “Rice” in English Wikipedia. Its Bengali counter part, that of “চাল” in Bengali Wikipedia is represented in Figure 6.2. It should be mentioned that Figure 6.2 represents almost the entire page in Bengali Wikipedia. Had the two languages been mapped using the system proposed in this thesis, using only a simple translation engine would have presented the entire data set of English Wikipedia in Bengali language. This

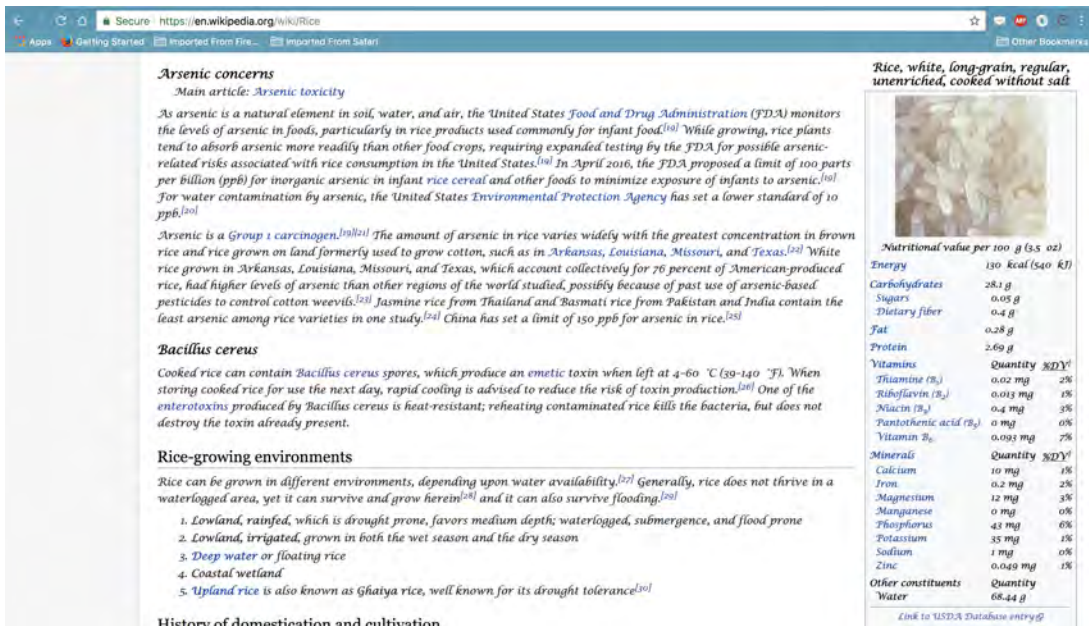


Figure 6.1: A sample English Wikipedia page.

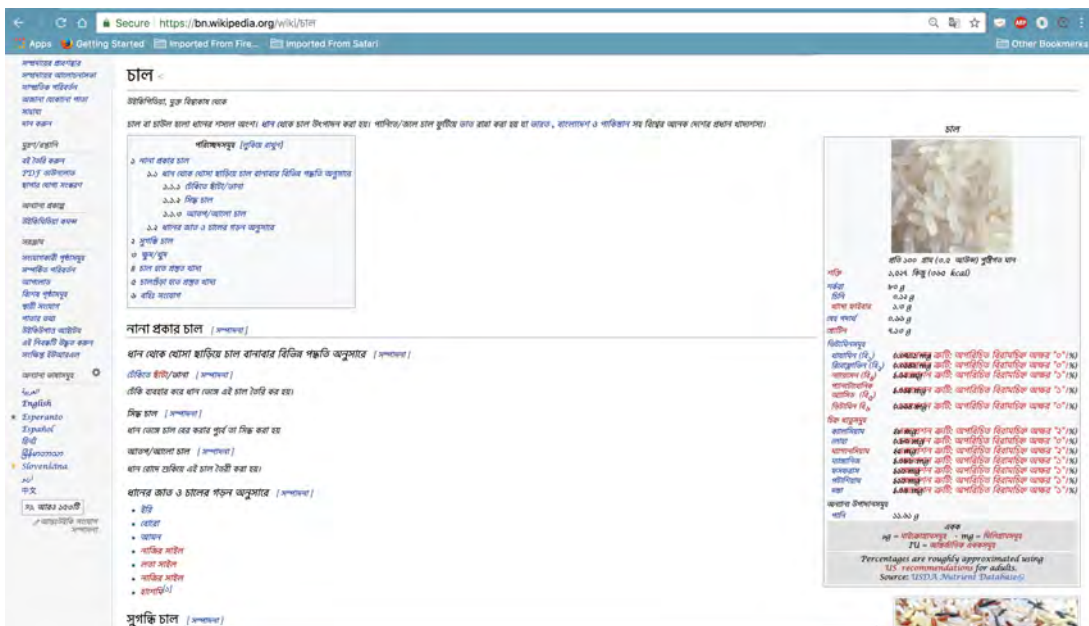


Figure 6.2: Bengali Wikipedia counter part to that of Figure 6.1

is how the proposed system is able to augment the data set of Wikipedia.

6.4 Gateway to Many More

On implementation of the Bengali DBpedia, it can be ensured that the system is fail-proof and thus, can be adapted for the implementation and integration of all the non-existent DBpedia. So it will be quite possible in future to expect a stable and standard knowledge repository which can be used by all web based systems as a reference for mapping. As new data are being added quite frequently, often it requires modification of the multilingual ontology. When considering cases of large-scale projects involving multilingual contributors from different countries, the requirement for change in the multilingual ontology may become mandatory [49]. Our proposed system presents to way to ensure minimum change in the ontology, but maximum efficiency by enhancing the data sources. This conclusion, thus, provides an unparalleled contribution on its own.

Finally, we can state that the outcome will be, any new data coming up in the world wide web will be machine readable and hence, Semantic Web will get its ultimate significance.

Appendix A

Bibliography

- [1] L. Yu, *A developer's guide to the semantic Web*. Springer Science & Business Media, 2011.
- [2] P. Hitzler, M. Krotzsch, and S. Rudolph, *Foundations of semantic web technologies*. CRC Press, 2009.
- [3] G. Antoniou and F. Van Harmelen, *A semantic web primer*. MIT press, 2004.
- [4] C. T. dos Santos, P. Quaresma, and R. Vieira, "A framework for multilingual ontology mapping," in *Proceedings of the International Conference on Language Resources and Evaluation, LREC*, ACM, 2008.
- [5] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. N. Mendes, S. Hellmann, M. Morsey, P. van Kleef, S. Auer, *et al.*, "DBpedia—a large-scale, multilingual knowledge base extracted from Wikipedia," *Semantic Web*, vol. 6, no. 2, pp. 167–195, 2015.
- [6] J. Guyot, S. Radhouani, and G. Falquet, "Ontology-based multilingual information retrieval.," in *CLEF (Working Notes)*, 2005.
- [7] M. T. M. Ankon, S. N. Tumpa, and M. M. Ali, "A multilingual ontology based framework for wikipedia entry augmentation," in *Computer and Information Technology (ICCIT), 2016 19th International Conference on*, pp. 541–545, IEEE, 2016.
- [8] F. Giunchiglia, M. Yatskevich, P. Avesani, and P. Shvaiko, "A large scale dataset for the evaluation of ontology matching systems," tech. rep., University of Trento, 2008.

- [9] J. Euzenat, C. Meilicke, H. Stuckenschmidt, P. Shvaiko, and C. Trojahn, "Ontology alignment evaluation initiative: six years of experience," in *Journal on data semantics XV*, pp. 158–192, Springer, 2011.
- [10] G. Rizzo, *Knowledge extraction from unstructured data and classification through distributed ontologies*. PhD thesis, Politecnico di Torino, 2012.
- [11] M. Manuja and D. Garg, "Semantic web mining of un-structured data: challenges and opportunities," *International Journal of Engineering (IJE)*, vol. 5, no. 3, p. 268, 2011.
- [12] M. T. M. Ankon and M. M. Ali, "Wikipedia entry augmentation by sub-merging entities based on multilingual ontology," in *Informatics, Electronics and Vision & 2017 7th International Symposium in Computational Medical and Health Technology (ICIEV-ISCMHT), 2017 6th International Conference on*, pp. 1–6, IEEE, 2017.
- [13] M. Morsey, J. Lehmann, S. Auer, C. Stadler, and S. Hellmann, "Dbpedia and the live extraction of structured data from wikipedia," *Program*, vol. 46, no. 2, pp. 157–181, 2012.
- [14] "Data set statistics." <http://wiki.dbpedia.org/services-resources/datasets/dataset-2015-04/dataset-2015-04-statistics>. [accessed 03-June-2017].
- [15] Wikipedia, "Rice — Wikipedia, the free encyclopedia." <http://en.wikipedia.org/w/index.php?title=Rice&oldid=723842664>, 2016. [Online; accessed 18-June-2017].
- [16] Wikipedia, "Help:Infobox — Wikipedia, the free encyclopedia." <http://en.wikipedia.org/w/index.php?title=Help%3AInfobox&oldid=719256926>, 2016. [Online; accessed 04-June-2017].
- [17] P. N. Mendes, M. Jakob, and C. Bizer, "Dbpedia: A multilingual cross-domain knowledge base.," in *LREC*, pp. 1813–1817, Citeseer, 2012.
- [18] H. K. Arnaout, *DBpediaSearch: an effective search engine for DBpedia*. PhD thesis, American University of Beirut, 2018.

- [19] J. Kuchar, “Augmenting a feature set of movies using linked open data.,” in *Challenge+ DC@ RuleML*, 2015.
- [20] D. Ritze, O. Lehmborg, and C. Bizer, “Matching html tables to dbpedia,” in *Proceedings of the 5th International Conference on Web Intelligence, Mining and Semantics*, p. 10, ACM, 2015.
- [21] K. Katzner, *The languages of the world*. Routledge, 2002.
- [22] J. Aitchison, *Language change: progress or decay?* Cambridge University Press, 2001.
- [23] dbpedia.org, “DBpedia 2016-04 Statistics.”
<http://wiki.dbpedia.org/dbpedia-2016-04-statistics>. [Online; accessed 17-June-2017].
- [24] Wikipedia, “Wikipedia — Wikipedia, the free encyclopedia.”
<https://en.wikipedia.org/wiki/Wikipedia>, 2018. [Online; accessed 17-April-2018].
- [25] Wikipedia, “Home — Wikipedia, the free encyclopedia.”
<https://www.wikipedia.org/>, 2018. [Online; accessed 17-April-2018].
- [26] “Semantic web.” <http://www.w3.org/standards/semanticweb/>. Last accessed 15 May 2017.
- [27] “Semantic web.”
http://http://www.semanticweb.org/wiki/Semantic_Web/. Last accessed 15 May 2017.
- [28] T. Berners-Lee and M. Fischetti, “Weaving the web. harpersanfrancisco. chapter 12,” tech. rep., ISBN 978-0-06-251587-2, 1999.
- [29] T. Berners-Lee, J. Hendler, and O. Lassila, “The semantic web,” *Scientific american*, vol. 284, no. 5, pp. 34–43, 2001.
- [30] R. V. Guha, “Light at the end of the tunnel,” in *Talk at the 12th International Semantic Web Conference (ISWC), Sydney*, vol. 10, 2013.

- [31] P. Cimiano, *Ontology Learning and Population from Text*, ch. 2, pp. 9–17. Springer US, 2006.
- [32] “Semantic web stack.”
https://en.wikipedia.org/wiki/Semantic_Web_Stacky, 2018.
[Online; accessed 30-May-2018”].
- [33] A. Maedche and S. Staab, “Ontology learning for the semantic web,” *IEEE Intelligent systems*, vol. 16, no. 2, pp. 72–79, 2001.
- [34] N. Choi, I.-Y. Song, and H. Han, “A survey on ontology mapping,” *ACM Sigmod Record*, vol. 35, no. 3, pp. 34–41, 2006.
- [35] M. Ehrig and Y. Sure, “Ontology mapping—an integrated approach,” in *European Semantic Web Symposium*, pp. 76–91, Springer, 2004.
- [36] J. Guyot, S. Radhouani, and G. Falquet, “Conceptual indexing for multilingual information retrieval,” in *Workshop of the Cross-Language Evaluation Forum for European Languages*, pp. 102–112, Springer, 2005.
- [37] “DBpedia — Wikipedia, the free encyclopedia.”
<https://en.wikipedia.org/wiki/DBpedia>, 2018. [Online; accessed 30-May-2018”].
- [38] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives, “Dbpedia: A nucleus for a web of open data,” in *The semantic web*, pp. 722–735, Springer, 2007.
- [39] Wikipedia, “Wikipedia:About — Wikipedia, the free encyclopedia.”
<http://en.wikipedia.org/w/index.php?title=Wikipedia%3AAbout&oldid=716776032>, 2016. [Online; accessed 03-June-2017].
- [40] A. Tawfik, F. Giunchiglia, and V. Maltese, “A collaborative platform for multilingual ontology development,” *World Academy of Science, Engineering and Technology, International Journal of Social, Behavioral, Educational, Economic, Business and Industrial Engineering*, vol. 8, no. 12, pp. 3795–3804, 2014.

- [41] R. C. Fernandez, E. Mansour, A. Qahtan, A. Elmagarmid, I. Ilyas, S. Madden, M. Ouzzani, M. Stonebraker, and N. Tang, “Sleeping semantics: Linking datasets using word embeddings for data discovery,” in *34th IEEE International Conference on Data Engineering, ICDE, Paris, France*, 2018.
- [42] J. G. Moreno, R. Besançon, R. Beaumont, E. D’hondt, A.-L. Ligozat, S. Rosset, X. Tannier, and B. Grau, “Combining word and entity embeddings for entity linking,” in *European Semantic Web Conference*, pp. 337–352, Springer, 2017.
- [43] T. R. Gruber, “A translation approach to portable ontology specifications,” *Knowledge acquisition*, vol. 5, no. 2, pp. 199–220, 1993.
- [44] “Ontology downloads 2016-10.”
<http://wiki.dbpedia.org/downloads-2016-10#ontology>, 2016.
[Online; accessed 08-May-2018”].
- [45] J. Samuel, “Towards understanding and improving multilingual collaborative ontology development in wikidata,” 2018.
- [46] Z. Szabó and D. Öri, “Information strategy challenges in the digital era how enterprise architecture management can support strategic is planning,” in *Software, Knowledge, Information Management and Applications (SKIMA), 2017 11th International Conference on*, pp. 1–8, IEEE, 2017.
- [47] A. Alba, A. Coden, A. L. Gentile, D. Gruhl, P. Ristoski, and S. Welch, “Multi-lingual concept extraction with linked data and human-in-the-loop,” in *Proceedings of the Knowledge Capture Conference*, p. 24, ACM, 2017.
- [48] G. Kobilarov, T. Scott, Y. Raimond, S. Oliver, C. Sizemore, M. Smethurst, C. Bizer, and R. Lee, “Media meets semantic web—how the bbc uses dbpedia and linked data to make connections,” in *The semantic web: research and applications*, pp. 723–737, Springer, 2009.
- [49] J. Samuel, “Collaborative approach to developing a multilingual ontology: A case study of wikidata,” in *Research Conference on Metadata and Semantics Research*, pp. 167–172, Springer, 2017.

Appendix B

Source Code

In this chapter, we put forth the raw code of the system implemented based on the research presented in the earlier sections.

B.1 System to Submerge Entities Based on Multilingual Ontology

```
/**
 *
 * @author TasnimAnkon
 */
55 public class SubmergingDataSets {

    static void submergeDataSet (String frenchEntity) {

        try {
            DetectorFactory.loadProfile (profileDirectory);
        } catch (LangDetectException ex) {
110         Logger.getLogger (Fetch_from_dbpedia.class.getName ())
                .log (Level.SEVERE, null, ex);
        }

        try {
```

```

115         ob.setName(frenchEntity);

           urlIdentifier = ob.getEnglishName();

           frenchUrlIdentifier = ob.getFrenchName();

       } catch (IOException ex) {

           Logger.getLogger(Fetch_from_dbpedia.class.getName())

120               .log(Level.SEVERE, null, ex);

       }

       String modelUrl = getEnglishModelURL(urlIdentifier);
       String objectUrl = getEnglishObjectURL(urlIdentifier);

125

       String frenchModelUrl = getFrenchModelURL(urlIdentifier);
       String frenchObjectUrl = getFrenchObjectURL(urlIdentifier);

       Model model = FileManager.get().loadModel(modelUrl);

       String queryString = "SELECT_?p_?o_" +

136           "WHERE_" +

           "_{" +

           "_____<" + objectUrl + ">_?p_?o_." +

           "_}";

141       Query query = QueryFactory.create(queryString);

       QueryExecution qexec = QueryExecutionFactory.create(query, model);

       englishProperty = new String[1000];
       englishValue = new String[1000];
       englishLanguage = new String[1000];

146       englishStatementType = new String[1000];

       englishSolutions = new QuerySolution[1000];

       try{

           ResultSet results = qexec.execSelect();

           while(results.hasNext()) {

151               QuerySolution soln = results.nextSolution();

```

```
RDFNode property = (Resource) soln.getResource("p");
RDFNode object = soln.get("o");
englishSolutions[numberOfProperties] = soln;
englishProperty[numberOfProperties] = property.asNode().
    getLocalName().toString();
156
    if (object != null) {
        String value = "";
        if (object.isLiteral()) {
            value = object.asLiteral().getLexicalForm();
161            englishLanguage[numberOfProperties] = object.
                asLiteral().getLanguage();
            englishStatementType[numberOfProperties] = "literal
                ";
        } else {
            value = object.asNode().getLocalName();
            if (value.isEmpty()) {
166                value = object.asNode().getURI();
            }
            englishStatementType[numberOfProperties] = "uri";
        }
        englishValue[numberOfProperties] = value;
171    }
    numberOfProperties++;
}
} finally{
    qexec.close();
176 }

181 Model FrenchModel = FileManager.get().loadModel(frenchModelUrl);
```

```

queryString = "SELECT_?p_?o_" +
              "WHERE_" +
              "_{" +
186         "_____<" + frenchObjectUrl + ">_?p_?o_" +
              "}_";

Query FrecnchQuery = QueryFactory.create(queryString);
QueryExecution FrecnchQExec = QueryExecutionFactory.create(
    FrecnchQuery, FrenchModel);
191 try{
    ResultSet results = FrecnchQExec.execSelect();
    frenchProperty = new String[1000];
    frenchValue = new String[1000];
    frenchLanguage = new String[1000];
196    frenchStatementType = new String[1000];
    frenchSolutions = new QuerySolution[1000];
    while(results.hasNext()) {
        QuerySolution soln = results.nextSolution();
        RDFNode property = (Resource) soln.getResource("p");
201    RDFNode object = soln.get("o");
        frenchSolutions[frenchProperties] = soln;
        frenchProperty[frenchProperties] = property.asNode().
            getLocalName().toString();

        if (object != null) {
206            String value = "";
            if (object.isLiteral()) {
                value = object.asLiteral().getLexicalForm();
                frenchLanguage[frenchProperties] = object.asLiteral
                    ().getLanguage();
                frenchStatementType[frenchProperties] = "literal";
211
                if (frenchLanguage[frenchProperties] == null ||

```

```
frenchLanguage[frenchProperties].isEmpty()) {  
    try {  
        detector = DetectorFactory.create();  
        text = value;  
216     if (text != null && !text.isEmpty()) {  
            if (isNumeric(text)) {  
                frenchLanguage[frenchProperties] =  
                    "en";  
            } else {  
                detector.append(text);  
221  
                ArrayList<Language> langlist =  
                    detector.getProbabilities();  
                for (int i = langlist.size(); i >  
                    0; i--) {  
                    if (langlist.get(langlist.size  
                        () - i).toString().contains(  
                            "fr")) {  
                        frenchLanguage[  
                            frenchProperties] = "fr"  
                        ;  
226     break;  
                    }  
                }  
            }  
        }  
231     } catch (LangDetectException ex) {  
        Logger.getLogger(Fetch_from_dbpedia.class.  
            getName()).log(Level.SEVERE, null, ex);  
    }  
    }  
    } else {  
236     try {
```

```
value = object.asNode().getLocalName();
detector = DetectorFactory.create();
text = value.toString();
if (text != null && !text.isEmpty()) {
241     text = text.replaceAll("_", " ");
    detector.append(text);

    ArrayList<Language> langlist = detector.
        getProbabilities();
for (int i = langlist.size(); i > 0; i--) {
246     if (langlist.get(langlist.size() - i).
        toString().contains("fr")) {
        frenchLanguage[frenchProperties] =
            "fr";
            break;
        }
    }
251 }
} catch (LangDetectException ex) {
    Logger.getLogger(Fetch_from_dbpedia.class.
        getName()).log(Level.SEVERE, null, ex);
}

256 if (value.isEmpty()) {
    value = object.asNode().getURI();

    if (frenchLanguage[frenchProperties] == null ||
        frenchLanguage[frenchProperties].isEmpty())
    {
        if (value.toString().contains("fr.dbpedia.
261     org")) {
        frenchLanguage[frenchProperties] = "fr"
        ;
    }
}
```

```

        } else if (value.toString().contains("fr.
            wikipedia.org")) {
                frenchLanguage[frenchProperties] = "fr"
                        ;
            }
        }
266     }

        frenchStatementType[frenchProperties] = "uri";
    }
    frenchValue[frenchProperties] = value;
271 }

        frenchProperties++;
    }
} finally{
276     qexec.close();
        System.out.println("Total_number_of_French_properties:_ " +
            String.valueOf(frenchProperties - 1));
}

for (i = 0; i < numberOfProperties; i++) {
    combinedProperty[i] = englishProperty[i];
    combinedValue[i] = englishValue[i];
    combinedLanguage[i] = englishLanguage[i];
301    combinedStatementType[i] = englishStatementType[i];

        combinedSolutions[i] = englishSolutions[i];
    }

//Semantic Rule 1
for (i = 0; i < frenchProperties; i++) {

```



```

    flag = false;
    iteratedSolution = frenchSolutions[i];
    iteratedProperty = frenchProperty[i];
    iteratedValue = frenchValue[i];
    iteratedLanguage = frenchLanguage[i];
    iteratedStatementType = frenchStatementType[i];
    for (j = 0; j < numberOfProperties; j++) {
        if (combinedProperty[j].equals(iteratedProperty)) {
            if (combinedValue[j].equals(iteratedValue)) {
                if (iteratedLanguage != null
                    && !iteratedLanguage.isEmpty()
                    && combinedLanguage[j] != null
                    && !combinedLanguage[j].isEmpty()) {
                    if (combinedLanguage[j].equals(iteratedLanguage
                        )) {
                        flag = true;
                        break;
                    }
                } else {
                    flag = true;
                    break;
                }
                flag = true;
                break;
            }
        }
    }

    //Semantic Rule 2
    if (iteratedLanguage != null && iteratedLanguage.equals("fr")
        && iteratedStatementType != null
        && iteratedStatementType.equals("literal")
        && !iteratedValue.isEmpty() && !isNumeric(iteratedValue
            )) {

```

```

    try {
        ob.setName(iteratedValue);
        temporaryValue = ob.getEnglishFromFrenchName();
346
        for (j = 0; j < numberOfProperties; j++) {
            if (combinedValue[j].equals(temporaryValue)) {
                iteratedProperty = combinedProperty[j];
                iteratedLanguage = "fr";
351
                System.out.println("Modified:_" +
                    iteratedProperty + "_" + iteratedValue + "_"
                    + iteratedLanguage);
                flag = false;
                break;
            }
        }
356
    } catch (IOException ex) {
        Logger.getLogger(Fetch_from_dbpedia.class.getName()).
            log(Level.SEVERE, null, ex);
    }
}

//Semantic Rule 3
if (iteratedLanguage != null && iteratedLanguage.equals("fr")
    && iteratedStatmentType != null
    && iteratedStatmentType.equals("uri")
365
    && !iteratedValue.isEmpty() && !isNumeric(iteratedValue
    )) {
    for (j = 0; j < numberOfProperties; j++) {
        if (combinedValue[j].equals(iteratedValue)
            && combinedStatementType[j] != null
            && combinedStatementType[j].equals("uri")) {
370
            if (combinedLanguage[j] == null || !
                combinedLanguage[j].equals("fr")) {

```

```
        iteratedProperty = combinedProperty[j];
        iteratedLanguage = "fr";
        flag = false;
        break;
375     }
    }
}

380 //Checking Redundancy
if (!flag) {
    combinedProperty[numberOfProperties + newProperties] =
        iteratedProperty;
    combinedValue[numberOfProperties + newProperties] =
        iteratedValue;
    combinedLanguage[numberOfProperties + newProperties] =
        iteratedLanguage;
385    combinedSolutions[numberOfProperties + newProperties] =
        iteratedSolution;
    newProperties++;
}
}

391 alphabeticalSort(numberOfProperties, newProperties,
    combinedProperty, combinedValue, combinedLanguage,
    combinedSolutions);

    generateXMLFile(urlIdentifier);
}

395 static void alphabeticalSort(int numberOfProperties, int newProperties,
    String[] combinedProperty, String[] combinedValue, String[]
```

```
        combinedLanguage,  
        QuerySolution[] combinedSolutions) {  
    String[] totalProperties = new String[numberOfProperties +  
        newProperties];  
400    String[] totalValues = new String[numberOfProperties +  
        newProperties];  
    String[] totalLanguages = new String[numberOfProperties +  
        newProperties];  
    QuerySolution[] totalSolutions = new QuerySolution[  
        numberOfProperties + newProperties];  
  
    int i, j;  
405  
    for (i = 0; i < (numberOfProperties + newProperties); i++) {  
        totalProperties[i] = combinedProperty[i];  
        totalValues[i] = combinedValue[i];  
        totalLanguages[i] = combinedLanguage[i];  
410        totalSolutions[i] = combinedSolutions[i];  
    }  
  
    ArrayIndexComparator comparator = new ArrayIndexComparator(  
        totalProperties);  
    Integer[] indexes = comparator.createIndexArray();  
415    Arrays.sort(indexes, comparator);  
  
    int traversingIndex;  
  
    finalProperties = new String[numberOfProperties + newProperties];  
420    finalValues = new String[numberOfProperties + newProperties];  
    finalLanguages = new String[numberOfProperties + newProperties];  
    finalSolutions = new QuerySolution[newProperties +  
        numberOfProperties];  
    int finalNumber = indexes.length;
```

```
425     for (i = 0; i < indexes.length; i++) {
        traversingIndex = indexes[i];
        finalProperties[i] = totalProperties[traversingIndex];
        finalValues[i] = totalValues[traversingIndex];
        finalLanguages[i] = totalLanguages[traversingIndex];
430     finalSolutions[i] = totalSolutions[traversingIndex];
    }

}

static void fetchLanguageOfURL(Integer[] indexes) {
    String language = "";
    String[] schemes = {"http","https"}; // DEFAULT schemes = "http", "
        https", "ftp"
    UrlValidator urlValidator = new UrlValidator(schemes);
447

    int i;

    for (i = 0; i < indexes.length; i++) {
        if(finalLanguages[i] == null && urlValidator.isValid(
            finalValues[i])) {
452         language = GetHTMLContent(finalValues[i]);
            finalLanguages[i] = language;
            System.out.println(language);
        }
    }
457 }

static int sortDataSet(Integer[] indexes) {
    int startValue = 0, finalValue = 0, consecutiveValue = 1;
    boolean referenceValue = true;
462    String referenceProperty = finalProperties[0];
```

```
    int[] contentLengths;

    int i, j;

467    String[] backupProperties = new String[finalProperties.length];
    String[] backupValues = new String[finalValues.length];
    String[] backupLanguages = new String[finalLanguages.length];
    QuerySolution[] backupSolutions = new QuerySolution[finalSolutions.
        length];

472    for (i = 0; i < finalProperties.length; i++) {
        backupProperties[i] = finalProperties[i];
        backupValues[i] = finalValues[i];
        backupLanguages[i] = finalLanguages[i];
        backupSolutions[i] = finalSolutions[i];
477    }

    int PropertiesBasedOnLength = 0;

    for (i = 1; i < indexes.length; i++) {
482        if (referenceProperty.equals(backupProperties[i])) {
            finalValue++;
            consecutiveValue++;
        } else {
            finalValue = i - 1;
487            referenceValue = false;
        }

        if (!referenceValue) {
            contentLengths = new int[consecutiveValue];

492

            for (j = 0; j < (consecutiveValue); j++) {
                contentLengths[j] = backupValues[j + startValue].length
```

```
        ( );
    }

    IntegerArrayComparator intComparator = new
        IntegerArrayComparator(contentLengths);
    Integer[] integerIndexes = intComparator.createIndexArray()
        ;
    Arrays.sort(integerIndexes, intComparator);

501

    int sortedTraversingIndex;

    for (j = 0; j < consecutiveValue; j++) {
        sortedTraversingIndex = integerIndexes[j];
506        finalProperties[startValue + j] = backupProperties[
            startValue + sortedTraversingIndex];
        finalValues[startValue + j] = backupValues[startValue +
            sortedTraversingIndex];
        finalLanguages[startValue + j] = backupLanguages[
            startValue + sortedTraversingIndex];
        finalSolutions[startValue + j] = backupSolutions[
            startValue + sortedTraversingIndex];
    }

511

    PropertiesBasedOnLength += consecutiveValue;

    startValue = i;
    finalValue = i;

516    consecutiveValue = 1;
    referenceValue = true;
    referenceProperty = backupProperties[i];
}

521    if (i == indexes.length - 1) {
```

```
contentLengths = new int[consecutiveValue];

for (j = 0; j < (consecutiveValue); j++) {
    contentLengths[j] = finalValues[j + startValue].length
        ();
526     }

IntegerArrayComparator intComparator = new
    IntegerArrayComparator(contentLengths);
Integer[] integerIndexes = intComparator.createIndexArray()
    ;
531 Arrays.sort(integerIndexes, intComparator);

int sortedTraversingIndex;

for (j = 0; j < consecutiveValue; j++) {
536     sortedTraversingIndex = integerIndexes[j];
    finalProperties[startValue + j] = backupProperties[
        startValue + sortedTraversingIndex];
    finalValues[startValue + j] = backupValues[startValue +
        sortedTraversingIndex];
    finalLanguages[startValue + j] = backupLanguages[
        startValue + sortedTraversingIndex];
    finalSolutions[startValue + j] = backupSolutions[
        startValue + sortedTraversingIndex];
541     }

    PropertiesBasedOnLength += consecutiveValue;
}
}

547

printPropertyList(indexes.length);
```



```
551     return PropertiesBasedOnLength;
    }

    public static String GetHTMLContent (String Url) {
        URL url;
556     InputStream is = null;
        BufferedReader br;
        String line;
        String htmlContent = "";

561     try {
            url = new URL(Url);
            is = url.openStream(); // throws an IOException
            br = new BufferedReader(new InputStreamReader(is));

566     while ((line = br.readLine()) != null) {
            htmlContent = htmlContent + line;
        }
    } catch (MalformedURLException mue) {
        mue.printStackTrace();
571     System.out.println("Malformed_URL");
        return "";
    } catch (IOException ioe) {
        ioe.printStackTrace();
        System.out.println("IO_Exception");
576     return "";
    } finally {
        try {
            if (is != null) is.close();
        } catch (IOException ioe) {
581     ioe.printStackTrace();
        System.out.println("2nd_IO_Exception");
        return "";
    }
}
```

```
    }
}

586
    org.jsoup.nodes.Document htmlDoc = Jsoup.parse(htmlContent);

    Element taglang = htmlDoc.select("html").first();

591    return taglang.attr("lang");
}

static void generateXMLFile(String fileName) {
    int i, j;
648    try {
        DocumentBuilderFactory docFactory = DocumentBuilderFactory.
            newInstance();
        docFactory.setNamespaceAware(true);
        DocumentBuilder docBuilder = docFactory.newDocumentBuilder();

        doc.appendChild(rootElement);

671        org.w3c.dom.Element firstLevel;
        Attr attr;

        for (i = 0; i < finalSolutions.length; i++) {
            firstLevel = doc.createElementNS(getNameSpace(
                finalSolutions[i].get("?p").toString()),
676                replaceWithNamespace(finalSolutions[i].get("?p").
                    toString()));

            if (finalLanguages[i] != null && !finalLanguages[i].isEmpty
                ()) {
                attr = doc.createAttribute("xml:lang");
                attr.setValue(finalLanguages[i]);
            }
        }
    }
}
```

```
681         firstLevel.setAttributeNode(attr);
        }

        if (conditionsForURIResource(i)) {
            attr = doc.createAttribute("rdf:resource");
686            attr.setValue(finalSolutions[i].get("?o").toString());
            firstLevel.setAttributeNode(attr);
        } else {
            firstLevel.appendChild(doc.createTextNode(finalValues[i
                ]));
        }
691        rootElement.appendChild(firstLevel);
    }

696    TransformerFactory transformerFactory = TransformerFactory.
        newInstance();
    Transformer transformer = transformerFactory.newTransformer();
    DOMSource source = new DOMSource(doc);
    StreamResult result = new StreamResult(new File(fileName + ".
        xml"));
    transformer.setOutputProperty(OutputKeys.INDENT, "yes");
701
    transformer.transform(source, result);

    System.out.println("File_saved!");

706 } catch (ParserConfigurationException pce) {
        pce.printStackTrace();
    } catch (TransformerException tfe) {
        tfe.printStackTrace();
    }
711 }
```