Keyword-based Information Retrieval over Enhanced Linked Data

RAHOMAN Md Mizanur

Doctor of Philosophy

Department of Informatics
School of Multidisciplinary Sciences
SOKENDAI (The Graduate University for Advanced Studies)

Keyword-based Information Retrieval over Enhanced Linked Data

Author:

RAHOMAN Md Mizanur

Supervisor:

Ryutaro Ichise

Doctor of Philosophy

September 2016

A dissertation progress report submitted to
the Department of Informatics,
School of Multidisciplinary Sciences,
SOKENDAI (The Graduate University for Advanced Studies)
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

Advisory Committee

Assoc. Prof.	Ryutaro Ichise	SOKENDAI
Prof.	Seiji Yamada	SOKENDAI
Prof.	Hideaki Takeda	SOKENDAI
Assoc. Prof.	Yusuke Miyao	SOKENDAI
Dr.	Tetsuya Nasukawa	IBM Research – Tokyo

Abstract

Linked Data are inference-enable, interlinked network data. They store knowledge with rich semantics. Currently Linked Data hold vast amount of knowledge, which are also growing rapidly. Success of these contemporary data depends on how effectively they can be used by the users and applications. Like other data, usage of these data primarily relies upon how easily they can be accessed, and how good the data are i.e., the quality of data. A good number of researches have been conducted to investigate these two issues, however, contemporary systems are still not good to tackle them effectively.

Considering that keyword-based query is an easy-to-use information access option, we find that contemporary systems are not effective to handle Linked Data's structural complexities. There are also very few systems that can handle specific semantics like "temporal semantics" – time and event related queries, however capturing them can leverage Linked Data usability. Since the contemporary systems suffer on handling those issues together, we propose Linked Data information access frameworks that are easy-to-use, effective to handle structural complexities, and facilitated to capture temporal semantics. We analyze structure of Linked Data and propose some defined templates for keywords, and their management to retrieve Linked Data information. While we capture temporal semantics by text analysis. On the other hand, we do not find many Linked Data quality assessment frameworks that can automatically identify all possible types of errors. Since manual quality assessment over Linked Data is not a feasible choice and data can hold different types of errors, an automatic quality assessment frameworks is a requirement. We adapt a novel unsupervised nearest-neighbor based outlier detection technique which is automatic, not susceptible to particular types of errors.

The proposed systems are easy to use and effective in their operations. They can support in leveraging Linked Data success.

To my family and friends.

Acknowledgements

I would like to express my sincere gratitude to my supervisor Professor Ryutaro Ichise for the continuous support of my study and research, for his patience, motivation, and immense knowledge. Without his guidance and persistent help this thesis would not have been possible. I appreciate all his contributions of time, ideas, and supervision to make my research experience productive and stimulating.

I would like to thank my committee members, Prof. Seiji Yamada, Prof. Hideaki Takeda, Assoc. Prof. Yusuke Miyao, and Dr. Tetsuya Nasukawa for their encouraging, insightful, and constructive comments.

I appreciate my lab members, Khai, Ball, Ebitsu, Lankesh, Lihua, Raul, and all the internship students, who supported me and helped me in my research.

Most importantly, none of this would have been possible without the love and patience of my wife - Silvia Islam and my son - Intishar Rahman. I would like to express my heart-felt gratitude to my parents, sisters who have been a constant source of love, concern, and support all these years.

Lastly, I thank the financial and educational support from Ministry of Education, Culture, Sports, Science & Technology (MEXT) and the Graduate University for Advanced Studies (SOKENDAI) during my studies. I appreciate all the incredible opportunities and memorable experience they offered so far.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this chapter, we briefly introduce background of our study (Section 1.1). Then we present motivation of this thesis, where we discuss problems or challenges (Section 1.2). We briefly introduce contributions of this thesis in Section 1.3. In the last Section, we outline each chapter of this thesis.

1.1 Background

Linked Data store knowledge in a network-like structure. Such structure can be compared with graph. Therefore, Linked Data are considered as semi-structure data [14]. Usually these data are presented with a machine understandable way. It significantly improves access and integration of such data. The inclusion of schema information or ontology information of data makes the Linked Data machine understandable, while the network-like structure of data helps to find potential link identification among various data, and construction of data links. Over any dataset, data schema usually describes data about data − meta data. In Linked Data perspective, this schema information is further extended to maintain relationship among the data or incorporate semantics over data which is called as data ontology. So, apart from describing meta data information, Linked Data ontology facilitates inference-enable data structure. Therefore, because of the inference-enable data structure of Linked Data, and their capability on data sharing, exposing, and connecting, Linked Data are consider as Semantic Web data [11] which Tim Berners-Lee, the visionary of Semantic Web, indicated as machine understandable data.

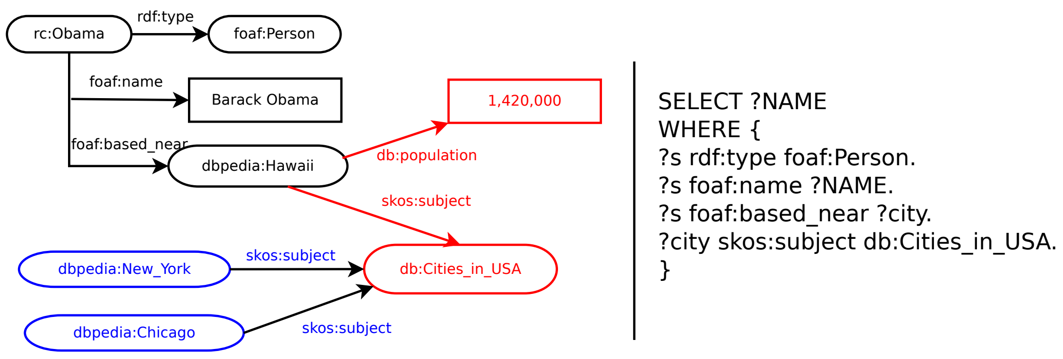
Usually, Linked Data are generated by two different methods. In first method, domain experts of a particular domain e.g., biology domain, medical domain, law domain etc. craft this kind of Linked Data, which heavily depend on manual activities. The example of such Linked Data are DrugBank[1], PubMed[2]. Because of manual intervention or expert intervention, this type of Linked Data are generally clean, but less frequent. In second method, pattern-based mapping, or rule-based mapping of source data extract Linked Data, which are generated either by automatic or semi-automatic Linked Data generation procedures. The example of source data could be Wikipedia text, automatically generated Sensor Data etc. On the other hand, the example of the Linked Data could be DBpedia[3], Freebase[4], LinkedSensorData[5] etc. Because of automatic generation of data, this type of Linked Data are more common, but potential to hold less cleaner data than manually generated Linked Data.

[1]https://datahub.io/dataset/fu-berlin-drugbank
[2]http://bio2rdf.org/
[3]http://dbpedia.org/About
[4]https://www.freebase.com/
[5]http://wiki.knoesis.org/index.php/LinkedSensorData

By principle, Linked Data adapt easy and simple data publishing strategies [10] – (1.) use Uniform Resource Identifiers (URIs) to denote things, (2.) use Hypertext Transfer Protocol (HTTP) URIs so that these things can be referred to and looked up ("dereferenced") by people and machine agents, (3.) provide useful information about the thing when its URI is dereferenced, leveraging the data query, and (4.) include links to other URIs, so that they can discover more things. By following the above strategies, Linked Data allow data publishers to publish their data with their own ontology. Although publishing data with publishers' own ontology perpetuates data heterogeneity, because of easy and simple data publishing strategies we see that Linked Data are growing rapidly. For example, rapid growth of Linked Open Data[6] can show this growing trend. While as of September 2011, Linked Open Data used to hold 295 datasets consisting of over 31 billion entries on various domains[7], and just within the next two and half years, as of April 2014, the datasets got increased to more than three folds and reached to 1014 datasets[8]. Therefore, we can assume that the Linked Data are growing rapidly. It indicates that currently Linked Data hold vast amounts of knowledge.

To access particular knowledge over the Linked Data, there exists some query frameworks such as SPARQL, RDQL etc. SPARQL is a structured query language (SQL) type expressive query (RQL, RDQL, or SPARQL [81]). However, data users need to know the data structure, the ontology and the query. Figure 1.1 shows example of Linked Data and SPARQL Query. The data part is shown on the left side while the query part is shown on the right side. The Linked Data adapt a graph-based model where each two nodes of data are connected with an edge or relation. The data are embedded with schema or ontology, e.g., "rdf:type" information is embedded for instance "rc:Obama". The SPARQL query shows that how we can query people (e.g., "Barack Obama") who live in "US Cities". More detail description of data, query etc. will be described in the following chapters.

[6]subset of Linked Data that are openly available to use
[7]http://www4.wiwiss.fu-berlin.de/lodcloud/state/
[8]http://linkeddatacatalog.dws.informatik.uni-mannheim.de/state/



```
SELECT ?NAME
WHERE {
?s rdf:type foaf:Person.
?s foaf:name ?NAME.
?s foaf:based_near ?city.
?city skos:subject db:Cities_in_USA.
}
```

FIGURE 1.1: Example of Linked Data and SPARQL query

1.2 Motivation

Linked Data currently hold a vast amount of knowledge. Therefore, the knowledge contained in Linked Data is worthy to investigate. We investigated Linked Data for two primary issues, which we understand that are crucial for Linked Data success [8]. They are:

(I.) **How Linked Data can be accessed in an easy and effective way by the general-purpose data users?**

(II.) **How Linked Data quality can be assessed automatically, irrespective of their error types, over a large Linked Dataset?**

Although there are ample amount of researches have been conducted over these two, we find that they are still open to do further researches. The below we describe motivational factors for both the issues in more details, respectively.

1.2.1 Information Access over Linked Data

The knowledge contained in Linked Data makes efficient data access options a necessity. However, information access over Linked Data is different from information access over other data. It is different from information access over traditional document-based data. It is also different from information access over usual graph-based data.

Comparing information access over Linked Data and document-based data, we find that while an information access over Linked Data system considers a query holistically, a traditional document-based information access system considers a query individually. That is why, an information access over Linked Data system needs to put all keywords of a query, in a semantic order, together so that it can retrieve more exact pitch of information that the query is searching for [7, 14]. On the other hand, a traditional document-based information access system retrieves relevant documents among the other documents, considering individual keywords of the query. In such a case, a document-based information access system might not need to consider the entire query, rather a part of the query can also produce required information need. Figure 1.2 shows information access comparison between over document-based data and Linked Data. It shows that the document-based information access system retrieves documents with different number of keywords (a.)), while the Linked Data information access system usually retrieves data for the entire query (b.)). In the figure, we see that both queries are posed to access information for keywords "spouse" and "Barack Obama". In a.), we find that the document-based information access system points individual document for each of the keywords. In this kind of access system, we generally can assume that a ranked list of such documents (shown in "dark blue" to "light blue") can provide us the required information need better, although there is possibility that any of them can serve the required information need. On the other hand, in b.), we find that the information access over Linked Data handles it differently. The information access over Linked Data system points Linked Data resources for all keywords of the query together. Although information access over Linked Data also ranks output of its query (shown in "dark blue" and "light blue"), we see that the information access over Linked Data always consider its query for all of the keywords holistically. So, we understand that simple use of document-based information access system will not fit over Linked Data.

Moreover, the information access over Linked Data is also different than the usual graph search because the traditional graph search may not be able to capture the rich semantics of Linked Data, that are presented with schema and ontologies.

Therefore, to access information over Linked Data for a query, a framework needs to find keywords. The keywords are important part of the query which hold query intension and can be linked to Linked Dataset. After finding the keywords, it needs to arrange keywords holistically so that the arrangement can simultaneously replicate subgraph of Linked Data graph and retrieve require information need.

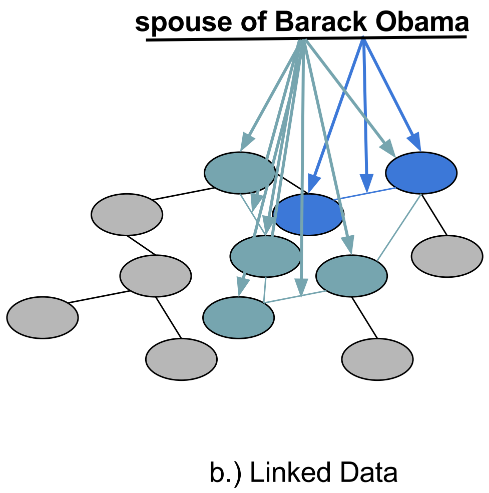


FIGURE 1.2: Basic difference between information access over document-based data and information access over Linked Data

We have found that information access over Linked Data is still an open research filed, because contemporary information access frameworks are still not effective. The below we outline some challenges / problems that motivated us to do our research.

1. Complex Competency – The information access over Linked Data requires complex user competency, from data modeling to data queries. For example, data users need to know the data structure, their vocabularies, data ontology, and SPARQL queries etc. These complexities get increased further because of Linked Data's data heterogeneity. For example, in Figure 1.1 Linked Data, we find that the data entries are described in three different vocabularies: foaf[9], db[10], skos[11]. Because of the heterogeneous nature of Linked Data, it is time consuming and practically infeasible to learn all them manually. As a result, accessing Linked Data is often not easy, especially for general-purpose users who have very little knowledge about the internal structure of Linked Data.

2. Costly Links – Information access over a Linked Data network requires following links [2, 5, 6, 42, 119]. However, simply following links over a large graph is costly, at least within a reasonable cost [2]. This is because, the link following can be considered as sub-graph searching, and searching a subgraph over a large graph is a subgraph isomorphism problem and is a

[9]http://www.foaf-project.org/
[10]http://dbpedia.org/ontology/
[11]https://www.w3.org/2009/08/skos-reference/skos.rdf

classical NP-complete problem. Most of the recent algorithms proposed to solve this problem apply structural features of graph and construct index, such as path, tree and subgraph. However, there is no solid theory foundation of which structure is the best one to construct the index. What is more, the cost of mining these structures is rather expensive [119, 120].

On the other hand, "templates" are investigated to search subgraphs over Linked Data graphs [95, 107]. The intuition behind the template use is that it would introduce a defined subgraph that supports finding of links and Linked Data endpoints over the Linked Data's big graph. However, the template-based Linked Data information retrieval studies have yet to provide concrete guidelines for template construction, ranking, and merging, all of which are required for effective adaptation of templates to Linked Data retrieval. For example, one recent research trend employed parse tree-based template construction [30, 31, 66, 107]. But the parser tree-based templates are not stable, and depending upon the language parser, we get multiple such trees, and wrong selection of tree lead to wrong or empty results. On the other hand, selection of all such possibilities are not be an efficient technique for retrieval of data from a large dataset [95]. Therefore, only linking of keywords towards the dataset is not enough to retrieve Linked Data information because keywords further need to fit into some structure or templates so that they can follow the Linked Data graph and retrieve the require information.

3. Temporal Lacking – The current information access framework over Linked Data lack of embedding temporal semantics [85]. However, temporal feature related information, such as a date and time or a time-specific event, is helpful for finding an appropriate result or for discovering a new relation [69, 93]. The temporal features filter out unnecessary data. For example, "US President during World War I" filter out all other US Presidents than "Woodrow Wilson".

   Though extraction of temporal feature related information of typical document-based data has quite a long history of research, the same kind of research for Linked Data has been comparatively less pursued [85].

Considering the above challenges, we find that information access over Linked Data often difficult, especially for general-purpose users who have very little knowledge about the internal structure of Linked Data, such as schema information or SPARQL query. Moreover, efficient query building technique over Linked Data graph is still not solved, because they require to follow the data links which are exponential in number. Therefore, we are motivated that an easy and efficient to use information access framework will exploit Linked Data that are growing rapidly. Thereby we can get closer towards the vision of Linked Data that machine and human both can access semantically aware information with lower complexity [8].

1.2.2 Assessment of Linked Data Quality

To use Linked Data effectively, Linked Data users commonly expect to use of high-quality data. This brings the need to identify erroneous data in the Linked Data. The below we outline the problem that motivated us to do the data quality assessment research.

1. Wrong Entries – The existing Linked Datasets are not always clean. Both Linked Data generation methods described in Section 1.1 can generate unclean data. In manually generated Linked Data, the erroneous data entries could be generated because of human errors. Moreover, data could be generated from multiple sources which sometime differ from one another. On the other hand, in automatically generated Linked Data, erroneous data entries could be extracted because of the wrong contents in source data. For example, Wikipedia contents could be wrong as they are built on people's mass contributions, who are not expert always. Furthermore, erroneous data entries could also be extracted because of wrong source-data mapping. For example, wrong InfoBox[12] mapping of Wikipedia contents extracts wrong entries. For both cases, DBpedia, which is extracted from Wikipedia text, data entries will contain wrong entries. For example, Table 1.1 shows some exemplary erroneous entries over DBpedia 3.8 dataset. The shaded rows hold erroneous entries.

   Although we do not have exact statistics that how good or bad the current Linked Datasets are, we show DBpedia statistics to get a rough idea of data. For example, Acosta et.al., and Zaveri et. al. calculated that DBpedia entries hold some kind of errors between 6.15% and 11.93% [1, 116]. These errors cover wide spectrum of error types such as value error, data type error,

[12]technique that is how Wikipedia keeps its data in some structure

TABLE 1.1: Exemplary Erroneous Entries over DBpedia 3.8 Dataset

Subject	**Predicate**	**Object**
Paprika	type	Book
Paprika	author	Yasutaka_Tsutsui
...	...	...
Freedom_in_Exile	type	Book
Freedom_in_Exile	author	14
...	...	...
Barack_Obama	type	Person
Barack_Obama	spouse	Michelle_Obama
...	...	...
Harry_Froboess	type	Person
Harry_Froboess	spouse	Germany
Harry_Froboess	spouse	Switzerland
...	...	...

class error etc. [1]. Likewise, Mendes et.al., has showed that at least 37% English DBpedia is not complete [74, 79].

Therefore, Linked Data are potential to contain errors [58, 87]. Because of Linked Data generation methods – both manual or automatic –, these errors can be any type of errors. However, currently we do not find any Linked Data quality assessment framework that can automatically assess the data for any error possibilities. This problem / challenge motivated us to investigate Linked Data for automatic quality assessment.

1.3 Contributions

In this thesis, we propose four frameworks BoTLRet (**B**inary Pr**o**gressive **T**emplate Paradigm over **L**inked Data **Ret**rieval) [83, 84, 86], TLDRet (**T**emporal **L**inked **D**ata **Ret**rieval framework) [85], LiCord (**L**anguage **i**ndependent **Co**ntent Wo**rd** Segmenter), and ALDErrD (**A**uto **L**inked **D**ata **Err**or **D**etector) [87] that help general-purpose users of Linked Data to access the information and assess the data quality.

To access Linked Data information, we proposed frameworks BoTLRet and TLDRet. BoTLRet is the basic framework, while TLDRet is an extension of BoTLRet to capture the temporal semantics. For a query, BoTLRet and TLDRet take keywords and extract required information need. The input keywords can be generated manually or automatically. In manual case, users craft keyword-based query by knowing the datasets' vocabularies. While, in automatic case, users can pose queries like natural language queries, later, the vocabulary specific keyword-based query can be constructed using state-of-the art tools such as language parser (e.g., Stanford Parser[13]), or machine learning-based keyword segmenter (e.g., LiCord [88]), and entity linker between keywords and vocabularies (e.g., [38]).

In this thesis, keywords are word segments that represent important sense of the sentence or query. In linguistics, such important word segments are mentioned as Content Words (CWs), and usually belong to nouns, most verbs, adjectives, and adverbs and refer to some object, action, or characteristic [112]. We define keywords that hold following properties i.e., the keywords

- contain one or more text segments with their word boundaries
- represent the natural language query intuitively

For example, for a natural language query "Who is Barack Obama is married to?", the keywords are {"Barack Obama" and "married"}. This is because, the important part of the query is "Barack Obama" and "married" and other parts "Who" "is" and "to" are not important because they are frequently used words which do not carry any special meaning. Therefore, "Barack Obama" and "married" intuitively mean the query.

On the other hand, to assess errors of Linked Data, we proposed Linked Data framework ALDErrD. In the below we describe our contribution for each of them, respectively.

1.3.1 Information Access over Linked Data

- **Proposal of a keyword-based information access framework over Linked Data**
  - **contribution in brief** – the proposal is a general-purpose information access framework over Linked Data that takes input as keywords, use

[13]http://nlp.stanford.edu/software/lex-parser.shtml

some templates to retrieve Linked Data information. The keywords are made out of natural language query, and they are posed according to the word-order of the query. To retrieve information over Linked Data, it needs to perform three tasks

(i.) decide keywords i.e., identification of important sense or part of (natural language) query called as CWs (Content Words)

(ii.) link keywords to datasets' vocabularies, which are data labels

(iii.) fit keywords to templates that retrieve the information

In this thesis, we contribute for tasks (i.) and (iii.) and keep task (ii.) as out-of-scope, which can be achieved by the state-of-the-art entity linking facilitates such as [38].

We contribute task (i.) by framework called LiCord that can identify CWs for the query. We describe it in Chapter 5.

On the other hand, we contribute task (iii.) by frameworks BoTLRet and TLDRet. The proposed frameworks addresse first two challenges on information access over Linked Data that is stated in Section 1.2.1 by providing a keyword-based information access facility. The framework uses some templates to retrieve Linked Data information. The template facilitated framework hides Linked Data's data structure complexities. It automatically embeds Linked Data schema, ontology etc, that usually lack in on a keyword-based Linked Data information accessing framework. Moreover, the templates support finding of links and Linked Data endpoints over the Linked Data's big graph, which is complex to follow. With the proposed framework, we provide concrete guidelines for template construction, ranking, and merging, all of which are required for effective adaptation of templates to Linked Data retrieval.

We describe them in Chapter 3 and 4, respectively.

- **Proposal of temporal feature related information access**
  - **contribution in brief** – the proposal is an extension of previous framework BoTLRet that facilitates temporal feature related information access over Linked Data. It addresses the third challenge on information access over Linked Data that we described in Section 1.2.1.

    The proposed framework is **TLDRet**. We describe it in Chapter 4.

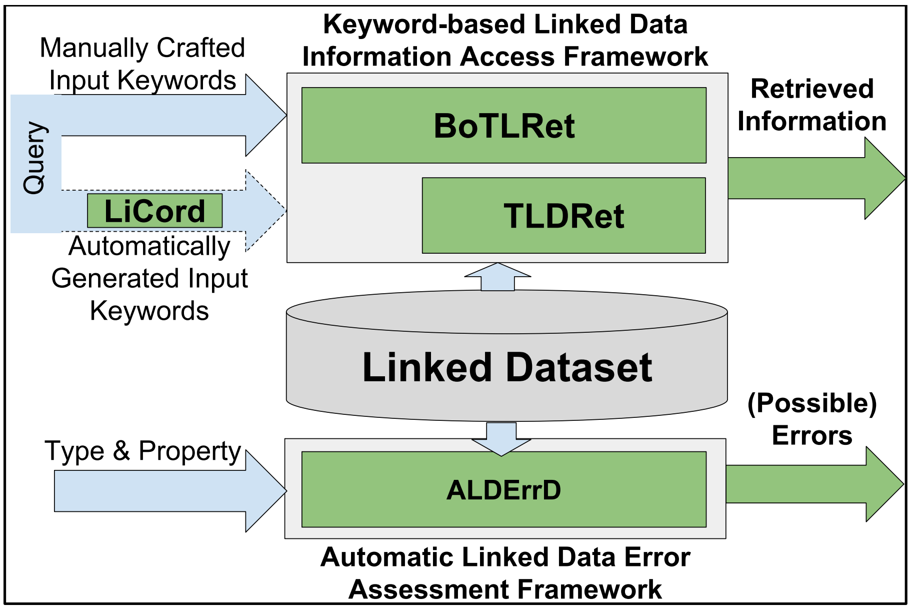

Keyword-based Linked Data Information Access Framework
Manually Crafted Input Keywords
Query
LiCord
Automatically Generated Input Keywords
BoTLRet
TLDRet
Retrieved Information
Linked Dataset
(Possible) Errors
Type & Property
ALDErrD
Automatic Linked Data Error Assessment Framework

FIGURE 1.3: Big picture of contributions. The "Green" parts are specific contributions.

1.3.2 Linked Data Quality Assessment

- **Proposal of an information assessment framework over Linked Data**
  - **contribution in brief** – the proposal is a framework to identify possible candidate of erroneous data over the type-annotated Linked Data. It addresses the problem on quality assessment over Linked Data that we described in Section 1.2.2.

    The proposed framework is **ALDErrD**. We describe it in Chapter 6.

Figure 1.3 shows "Big Picture" of our contribution. It shows that we contributed for two primary issues, keyword-based information access over Linked Data, and automatic Linked Data assessment. The "Green" parts are our contributions. In automatic keyword generation, we contributed part of it i.e., the CWs finding (by the framework LiCord), which is shown in dotted mark.

1.4 Outline

The remainder of this paper is divided as follows: In Chapter 2, we describe more detail background that are essential for this thesis, followed by works related to our study. In Chapter 3, we describe the basic information access framework over Linked Data. In Chapter 4, we extend the basic framework to capture temporal feature related Linked Data information. In Chapter 5, we show a technique that can be used to automatically devise keywords for the query. In Chapter 6, we describe the Linked Data quality assessment framework. In Chapter 7, we discuss on our contributions. Finally, Chapter 8 concludes our work.

Chapter 2

Background

In this chapter, we introduce more detail background knowledge on Linked Data and their technologies (Section 2.1). Later we discuss on information access research and their related works (Section 2.2). Next, like information access research, we discuss on data quality assessment and their related works (Section 2.3). For both research topics, we list some unsolved problems or challenges in the related works that we will solve in this thesis. Finally we summarize this chapter.

2.1 Linked Data Technologies

In this section, we will describe Linked Data and their associated technologies. It includes Linked Data, Linked Data vision, data publishing principles, data model, data schema, ontologies, data query etc. The background knowledge on Linked Data will help us to go through with this thesis.

2.1.1 Linked Data

Linked Data are data that published on the Web in such a way that they are linked by network links, preserved by their inference-enable meanings, readable and explorable by machines, and can in turn be linked to from external data sets [14]. Tim Berners-Lee, the visionary of Linked Data, indicated that Linked Data should be presented with a machine understandable manner. He predicated that it significantly will improve Linked Data access and their integration. Therefore, the Linked Data initiative has opened new opportunities in data usage, where this network-like structure of the data are considered as potential for link construction and identification among various data [14, 98].

Tim Berners-Lee, also the inventor of the World Wide Web, introduced the Linked Data principles [10]. He suggested to adapt Linked Data with three major technologies: Uniform Resource Identifier (URI), which is a generic means to identify entities or concepts in the world; Hypertext Transfer Protocol (HTTP), which is a simple yet universal mechanism for retrieving resources or descriptions of resources, and; Resource Description Framework (RDF), which is a generic graph-based data model for structuring and linking data that describe things in the world. According to Tim Berners-Lee, the best practices of Linked Data handling are [46]:

- Use URIs to denote things.
- Use HTTP URIs so that these things can be referred to and looked up ("dereferenced") by people and user agents.
- Provide useful information about the thing when its URI is dereferenced, leveraging standards such as RDF* and SPARQL.
- Include links to other URIs, so that they can discover more things.

Figure 1.1 shows example of Linked Data and the SPARQL query code snippet that how data can be queried. We will describe the Figure more details in the following sections.

2.1.2 Data Model

The data model that Linked Data adapts is called Resource Description Framework (RDF). It is used to interchange data on the Web. According to World Wide Web organization[1], RDF supports in data merging even if the underlying schema differ. Therefore, the data consumers of Linked Data do not need to change their data, even the RDF schema get evolved over the period.

RDF data model extends the Web linking structure. Its basic constituent is referred to as a "triple" which consists of two ends of the link, connected from source node to the destination node. In a triple, the two nodes are considered as "subject" and "object", and the link or edge is considered as "predicate", therefore, triples are form with <subject, predicate, object> expressions [14].

The linking structure of RDF forms a directed, labeled graph. In RDF data model, the edges represent the named link between two resources, represented by the graph nodes. Therefore, the RDF data model can be visualized with the graph view of network-data. This is the easiest possible mental model for RDF. It allows storing both structured and semi-structured data. RDF data model facilitates data mixing , data exposing, and data sharing across different applications, which is primary vision Linked Data or Semantic Web data [11]. So, the network-like structure of Linked Data opens the potentiality of new data discovery because they can be accessed by following the network links [5, 6, 42].

Figure 1.1 visualizes Linked Data on the left side. We see that the Linked Data adapt a graph-based model where each two nodes i.e., subject or object (as an instance or a literal value) of data are connected with an edge or relation or predicate. We also see that Linked Data keep data in more details and more linked ways. For example, instance "rc:Obama" in Figure 1.1 explicitly keep three triples: <rc:Obama, rdf:type, foaf:Person>, <rc:Obama, foaf:name, Barack Obama>, and <rc:Obama, foaf:based_near, dbpedia:Hawaii>, which are more detail representation of data than that of usual document-based data. We also see that resource

[1]https://www.w3.org/RDF/

"db:Cities_in_USA" is connected to three different datasets, considering the three different color datasets are coming from three different data sources, which are not common in document-based data. The more linked and more detail presentation of Linked Data support machines to explore the data efficiently.

On the other hand, in Linked Data the storage paradigm deviates from traditional repository-centric infrastructures to an open publishing model that allows other applications to access and interpret stored data. Unlike other constraint-based data such as relational data, data publishing over Linked Data is rather easy, because, by maintaining Linked Data principles, individual data publishers can publish their data with their own data schema without knowing other publishers' data schema. Thus, either the same data publishers or another group of publishers can connect published data and construct global data.

2.1.3 Data schema and Ontology

Linked Data are usually published with schema or ontology. Over any database, data schema usually describes data about data – the meta data – which defines data structure. In Linked Data perspective, this schema information is further extended to maintain relationship among the data or incorporating semantics over data which is called as data ontology. So, apart from describing meta data information, Linked Data ontology facilitates inference-enable data structure – using classes and properties – which supports automatic reasoning. As mentioned before, data publishing over Linked Data is maintained by individual data publishers which advocates that there is no need to use any specific ontology. Although it is good practice to use already established ontology since it gives easy integration opportunity [10, 46], practically Linked Data adapt this loose data publishing strategy to foster the rapid population of data considering that data will be self-describing, and thereby they can be looked up and reused. In Linked Data population, allowing data publishers to use their own ontology perpetuates data heterogeneity. However, it boots data generation of Linked Data. With this consequence we see that Linked Data cloud, as of September 2011, used to hold 295 datasets consisting of over 31 billion resource document framework (RDF) triples on various domains which have become interlinked by around 504 million RDF links[2]. While within the next two and half years the datasets got increased to

[2]http://www4.wiwiss.fu-berlin.de/lodcloud/state/

more than three folds, and, as of April 2014, it reached to 1014 datasets[3]. So, these days such Linked Data hold vast amounts of knowledge. Figure 1.1 shows how we embed schema or ontology on Linked Data. For example, “rdf:type” information is embedded for instance “rc:Obama”, by which it is understood that “rc:Obama” is a “Person”.

2.1.4 Query

Usually Linked Data framework offers SPARQL Protocol and RDF Query Language (SPARQL), which is a powerful RDF query language that enables data users to access to Linked Data [46]. SPARQL can be used to express queries across diverse data sources, where the data is stored natively as RDF or viewed as RDF via middle-ware [41]. Usually the users of Linked Data require to know the data model, the data ontology and the structured query language (SQL) type expressive queries (RQL, RDQL, or SPARQL [81]) For example, Figure 1.1 shows on its right side that how SPARQL query can generate people (e.g., “Barack Obama”) who live in “US Cities”.

2.2 Information Access over Linked Data

Linked Data currently hold a vast amount of knowledge, which is also growing rapidly. Therefore, the knowledge contained in Linked Data is worthy to investigate. As mentioned in Chapter 1, we investigate Linked Data for two primary issues, (1.) Information Access, and (2.) Data Quality Assessment. These two issues are crucial for success of Linked Data [8]. Here we will briefly outline about information access technique, its contemporary systems and current challenges and their possible solutions. The same will be outlined for quality assessment in next Section (2.3) .

In Chapter 1, we already discussed that information access over Linked Data is different from information access over usual document-based data. While an information access over Linked Data system considers a query holistically, a traditional document-based information access system considers a query individually (described in Section 1.2.1). Therefore, a simple replacement of document-based

[3]http://linkeddatacatalog.dws.informatik.uni-mannheim.de/state/

information access system does fit for Linked Data information access. On the other hand, as discussed earlier in this Chapter, since Linked Data is a graph-based data, we generally can think that traditional graph-search can access the data. However, because of Linked Data's rich semantics, that are often hidden in the complex structure and attributes in the form of Linked Data schema, ontology, vocabularies etc., the traditional graph-search is not well fit for Linked Data information access [52].

Usually, a Linked Data information access framework needs to find keywords for the query. As we defined the keywords in Chapter 1 Section 1.3, the keywords are important part of the query which hold query intension and can be used to link the Linked Dataset. Then, we arrange keywords holistically so that the arrangement can replicate subgraph of Linked Data graph according to Linked Data semantics and retrieve the require information need.

Lately there proposed multiple graph-search algorithms that are focused to capture complex structure and attributes of a graph, which Khan et. al. [52] categorized into three different graph-search algorithms:

(a.) Mining Query Algorithms, which are to find all frequent subgraphs and patterns from a large scale graph or a set of graphs. For example, gSpan [114], AGM [48], graph pattern matching algorithms like [23, 24, 60], or, top-k proximity pattern mining [53] etc. belong to this category.

(b.) Matching Query Algorithms, which are to find a given query graph or pattern from a target network. For example, graph pattern matching algorithms like [18, 24, 26, 101] etc. belong to this category.

(c.) Selection Query Algorithms. which are to identify the top-k nodes in a target network based on various input criteria. For example, ranked keyword-based graph search algorithms [12, 44, 100] or, skyline [15], SimRank [65] etc. belong to this category.

    The basic difference between the Matching Query Algorithms and Selection Query Algorithms is that, the Matching Query Algorithms have explicit structures (e.g. query graphs); whereas the structure is implicit in the Selection Query Algorithms (e.g. keyword search).

Our literature review on Linked Data information access showed that some of the contemporary systems, not all, utilized those graph-search algorithms to capture the semantics, but they are still expensive [119, 120]. The below we outline the related works on information access over Linked Data, and show, if appropriate, which graph-search algorithms they used from the above category. The literature review will reveal the exact challenges that the contemporary Linked Data information access systems still suffer, as “unsolved issues”. Later, we briefly outline our proposed solutions to solve the unsolved issues. The detail description of our proposal are described in Chapter 3 and 4.

2.2.1 Related Works

To address the challenge of Linked Data information access (stated in Chapter 1 Section 1.2.1) we review the contemporary systems that deal with Linked Data information access. With our literature review, we categorize them into six categories: (1.) Look-up-based Systems, (2.) Easy Query Building or Visualizing Systems, (3.) Supervised Machine Learning-based Systems, (4.) Template-based Systems, (5.) Pivot Point-based Systems and (6.) Temporal Information Access Systems.

The Look-up-based Systems are similar to traditional document-based information access systems. On the other hand, the Easy Query Building or Visualizing Systems and the Pivot Point-based Systems hold some information access systems that users need to construct their queries or guide for the queries. We review them to address the easy information access challenge. The Supervised Machine Learning-based Systems use some training examples to construct the query. While the Template-based Systems are language tool-based systems that use some kind of template to access information, which usually are automatic or semi-automatic. The Template-based Systems, by introducing the templates, try to minimize Linked Data’s structure-related complexity such as data link following, ontology understanding etc. We review them to address the effective information access that can hide Linked Data complexity. Finally, the Temporal Information Access Systems are used to retrieve temporal feature related information. We review them to address the temporal semantics embedding challenge. The below we discuss them more details.

(1.) **Look-up-based Systems** – look-up, index or crawl for RDF resources either for one data source or heterogeneous data source. Systems like Sindice [106]

Falcon [25], Swoogle [33] etc. belong to this category. These are the early Linked Data information access systems.

The below we briefly discuss them.

- *Sindice* [106] is a lookup-based information access system that construct index over resources, crawl on the Semantic Web. It consists of several independent components which are pipe-lined to get crawling, indexing, and querying. The crawler autonomously extracts RDF data from the Web data. Then, it adds to a indexing queue. The index allows applications to automatically retrieve sources with information about a given resource. It also allows resource retrieval through inverse-functional properties, which also offers full-text search and index SPARQL endpoints. Therefore, Sindice only acts as locator of RDF resources, returning pointers to remote data sources, and not as a query engine.

  Sindice actually is not an end-user application. However, its service can be used by any decentralized Semantic Web client application. It facilitates to locate relevant data sources. Therefore, Sindice is more close to standard Web search engines. However, it focus is specific towards Semantic Web concepts, procedures and metrics, rather than to Semantic Web search engines. Therefore, Sindice is not an information access system over Linked Data or Semantic Web data that retrieves it data holistically, rather it retrieves data much like document-based information access. So, we can say that Sindice aims at providing general query capabilities over the collections of all the Semantic Web data or Linked Data.

- *Falcon* [25] supports users with the option of accessing Web Resources for objects, concepts and documents. It provides very simple query capabilities. The object search is fitted to look for concrete items like people, places etc. The concept search is related to search for classes and properties that are available in ontologies published on the Web. The document search feature is suited a more traditional search engine experience, where search results point to RDF resources that contain the specified search terms.

  It is to be mentioned that, in Falcon RDF resources may be considered as distinct entities. While, the document Web and the data Web form one connected, navigable information space. In this way, Falcon gives, a navigation opportunity. For example, a user may perform a search in the existing document Web, follow a link from an HTML document into the Web of Data, navigate this space for some time, and then follow a link to a different HTML document, and so on.

Therefore, Falcon fits more towards traditional search engine-like information access, than the Linked Data required holistic data search.

- *Swoogle* [33] is also crawler based information access system. It index the Semantic Web data and retrieve them with the index. Swoogle also store meta-data of Semantic Web data, and calculate relationships between the documents. The retrieval input either character N-Gram or URI.

  Therefore, Swoogle also fits more towards traditional search engine-like information access, than the Linked Data required holistic data search.

(2.) **Easy Query Building or Visualizing Systems** – propose easy query building techniques that produce required information need or visualize the data. Systems like GoRelations [40], SPARQL Views [27], DERI Pipes [80], KOIOS [13], Hermes [104] etc. belong to this category. This category of systems still require know-how of Linked Data basics, sometime even in-depth knowledge of one of more datasets e.g., in Hermes.

This kind of systems utilize the Match Query Algorithms or Selection Query Algorithms to find the subgraph over Linked Data graph.

The below we briefly describe two of them.

- *GoRelations* [40] offers a query building technique that users need to devise to retrieve data from Linked Data graph. GoRelations (Graph of Relations) is an open domain question answering system, which authors claim is easy to learn and use.

  GoRelations consists of two components: a semantic graph interface (SGI) and an automatic translator mapping. With SGI, user can construct an intuitive query. In query, user needs to define relational part, and instance and concept part of the query. Then it automatically builds a semantic graph by connecting relations with instances and concepts. On the other hand, the automatic translator mapping find each entity by name or value and its concept in the query context. Thereby, GoRelations maps input query terms in the semantic graph which fits ontology terms. Finally, it generates a SPARQL query from the mappings.

- *DERI Pipes* [80] supports in Semantic Web data mash-ups. It can preserve desirable properties of data. Example of such properties are abstraction, encapsulation, component-orientation, code re-usability and maintainability etc.

  In DERI Pipes, it extracts data by using several pipes or filters. The filtered data are retrieved from existing Web contents

(3.) **Supervised Machine Learning-based Systems** – take supervised training data, and use user feedback to refine the Linked Data query. Systems like MQLOD [72], AutoSPARQL [62], [103] etc belong to this category. These category of systems depends upon machine learning outputs which are heavily depends upon training data. In real world case, accumulating these training data is not easy.

The below we briefly describe one of them.

- *AutoSPARQL* [62] uses supervised machine learning in its query modeling. In training, it allows users to ask queries without knowing the schema of the underlying knowledge base and without being expert in the SPARQL query. That is, it engages users in active supervised machine learning to generate a SPARQL query. The positive training examples are generated for resources that should be found for SPARQL query, and the negative training examples are generated for resources that should not be found for a SPARQL query.

  The user can either start with a question as like other QA systems. Or, by directly searching for a relevant resource as directory search, where each search results generates two sets of resources, positive resources and negative resources. For example, user searches for “Berlin”, s/he then selects an appropriate result, which becomes the first positive example. After that, s/he is asked a series of questions on whether a resource, e.g. “Paris”, should also be contained in the result set. These questions are answered by “yes” or “no”. This feedback allows the supervised learning method to gradually learn which query the user is likely interested in.

(4.) **Template-based Systems** – use templates to fit in some part of Linked Data graphs. Systems like PowerAqua [66], TBSL [107], FREyA [30, 31], SemSek [4], CASIA [45], DEQA [63] GraphPattern [95] etc. belong to this category. Various techniques are used to create the templates. One technique is using of natural language tools such as language parser over the input query. In this technique, the parser outputs, which generally are parse trees, which are considered as templates and are used to predict the possible subgraph (over Linked Data graph). A parse tree is a tree with annotated[4] part of input query. Therefore, the templates are considered as subgraph of Linked Data graph. However, when templates are constructed for input keywords, the contemporary systems fail to manage them effectively because most of them used language-based tools such as language parser which are not stable, and are not well defined [30, 31, 66, 107]. As a consequence, the contemporary systems sometime construct wrong templates which eventually generate wrong or empty results. On the other hand, selection of all such possibilities are not be an efficient technique for retrieval of data over a large dataset [95].

This kind of systems utilize the Match Query Algorithms or Selection Query Algorithms to find the subgraph over Linked Data graph.

This category Linked information access is popular with natural language or keyword-based query [77, 91]. Moreover, the template can easily capture the Linked Data semantics. Since this technique is popular recently and we also will use templates in our proposal, we briefly discuss five of them from this category.

- *PowerAqua* [66] performs question answering over structured data on the fly. It is an open domain question answering system, which is agnostic towards particular vocabulary or structure of the dataset.

  PowerAqua follows a pipeline architecture. Here input query is first transformed into query triples of the form subject, property and object by means language parser. Then, suitable query triples are mapped that identifies Linked Data resources. In mapping, it uses various language resources such as a WordNet search in order to find synonyms, hypernyms, derived words and meronyms. With the mapping, a set of ontology triples that jointly cover the user query is derived. Finally,

[4]annotation could be POS tagging, NER tagging etc.

since it could be happened that resulting triples may lead to only partial answers, all results are combined into a complete answer.

The performance of PowerAqua heavily depends on accuracy of language processing tools.

- *FREyA* [30, 31] allows users to enter queries in keyword-based query or natural language-based query. FREyA adapts a semi-supervised query technique.

  Like PowerAqua[66], it also follows a pipeline architecture, and generates query outputs. At first, it generates a syntactic parse tree in order to identify the answer type. Then, it starts with a lookup, annotating query terms with ontology concepts. If the lookup finds ambiguous annotations, the user is engaged to clarify the correct annotation. In such a case, the user's selections are saved and used for training the system. It improves system's performance over the time. Next, on the basis of the ontological mappings, system generates triples. Generation of triples are considered with domain and range of the properties. Finally the resulting triples are combined to generate a SPARQL query.

  The performance of FREyA is also heavily depends on accuracy of language processing tools. Moreover, it performs as a semi-supervised system.

- *QAKiS* [20] is a question answering system over DBpedia. Its main technique relies on filling the gap between natural language text and labels of ontology concepts. To fill the gap, it uses WikiFramework repository. The WikiFramework repository is automatically built by extracting relational patterns from Wikipedia free text. Authors specified WikiFramework for properties of DBpedia ontology. For example, a natural language pattern that expresses the relation *birthDate* is *born on*.

  QAKiS mainly focuses simple questions that contain one named entity and is connected to the answer via one relation. We see that QAKiS first determines the answer type as well as the type of the named entity, and next matches the resulting typed question with the patterns in the WikiFramework repository, and finds the most likely relations, which is then used to build a SPARQL query.

The building of WikiFramework repository heavily depends on language processing tools. Therefore, the performance of QAKiS heavily depends on quality of WikiFramework repository.

- *SemSek* [4] is a question answering system that focuses on matching natural language expressions to ontology concepts.

  SemSek has three steps: linguistic analysis, query annotation, and a semantic similarity measure. The query annotation step mainly looks for entities and classes in a DBpedia index that match the expressions occurring in the natural language question. The natural expression matching part is guided by the syntactic parse tree, provided by the linguistic tools like language parser. SemSek retrieves a ranked list of terms following the dependency tree. In order to match these terms to DBpedia concepts, SemSek uses two semantic similarity measures, first one is explicit semantic analysis based on Wikipedia, and the second one is semantic relatedness measure based on WordNet structures.

  Therefore, we see that the performance SemSek thus mainly relies on semantic relatedness of query.

- *TBSL* [107] is a question answering system that focuses on transforming natural language questions into SPARQL queries.

  It first parsed natural language questions with language parser, which generates several parsed trees. Among the parsed trees, it tries to find the best best representation of the semantic structure of the query question. These trees are considered as templates for the query.

  Therefore, TBSL first produces parser output-based templates that tries to understand the internal structure of the question. Then these templates are mapped with Linked data RDF resources. The mapping is done with the help of statistical entity identification and predicate detection. Finally, the templates are converted to SPARQL queries.

  Template creation of TBSL is based on natural language tools. Therefore, performance of TBSL heavily rely upon the performance of the tools.

(5.) **Pivot Point-based Systems** – track a "pivot point" (i.e., particular part or point of input query) and explore the remaining points, and thereby determine all required points (i.e., subgraph) of Linked Data graph. Systems like Treo [35], NLP-Reduce[51], QUICK [117] belong to this category of systems.

Here systems use single pivot point which leads next exploration points. However, inappropriate selection of pivot point will affect system's performance. Moreover, this approach also could miss contextual information attachments among the points, or could predict a subgraph that generates empty result. Dynamic programming based subgraph prediction, which was investigated in Treo, could be a possible workaround. However, in such a case, backtracking and picking of another point increases the data access complexity. This is because, if such approach is adapted, each of this backtracking needs to check for all the instances that correspond to the point. Furthermore, in real world scenario, every point (i.e., keyword) corresponds to multiple instances. For example, with exact string matching, keyword "Germany" has at least 22 instances over DBpedia 3.8. This also increases data links following complexity.

Here, all require points retrieve subgraph over Linked Data graph. Therefore, most of the cases, the subgraph finding belong to Selection Query Algorithms.

The below we briefly discuss two of them.

- *QUICK* [117] works on pre-defined domain specific ontologies. It works as exploring Linked Data resource by pivoting some resources.

  In QUICK, user needs to put her or his query in keywords. The system then guides the user through an incremental construction process. This incremental process ultimately leads the user to the desired semantic query. Here QUICK considers that user possesses basic domain knowledge, although it does not need to know specific details of the ontology, or proficiency in a query language. In this way, QUICK combines the familiar keyword-based query towards the semantic queries i.e., the holistic query.

- *Treo* [35] adapts a kind of on-line navigation-based graph-search which traverses dereferenced URIs to navigate the Linked Data graph. It also works as exploring Linked Data resource by pivoting some resources.

  The query processing is stated by determining pivot entities in the natural language question. The pivot entities can be mapped to instances or classes. We can consider them as an entry point, from where next navigation will be populated to search in the remaining query. Starting from these pivot points, Treo navigates through the neighboring nodes, computing the semantic relatedness between query terms and vocabulary terms, which ultimately lead the the exploration process. By this way, it returns a set of ranked triple paths from the pivot point to the final resource representing the answer. Here ranking is done by the average of the relatedness scores over each triple path.

Since pivot point give next exploration points, the performance of Treo heavily depends on correct identification of pivot point.

(6.) **Temporal Information Access Systems** – Temporal information access systems are relatively less in practice. In 2011, a study by Vandenbussche et al. at the Detection, Representation, and Exploitation of Events in the Semantic Web (DeRiVE) workshop advocated an initiative regarding the extraction of temporal feature related Linked Data ([109]). The study by Vandenbussche et al. [109] focused primarily on image extraction. Khrouf et al. and Troncy et al. both showed *event* related ontology integration, i.e., what (event), when (date/time), where (geo) and who (participant of an event). But these studies considered that Linked Data would be presented with a fixed ontology (namely, LODE) and they ignored the possibility of accessing other temporal ontologies ([54, 105]).

To our knowledge, no information access system over Linked Data that can retrieve Linked Data information for rich temporal semantics, such as those given in natural language query or keyword query. We understand that the natural language query can include rich temporal semantics. We will propose such system in this thesis.

2.2.2 Unsolved Issue

To access information over Linked Data, in Section 1.2.1, we addressed three different challenges or problems – (1.) Complex Competency, (2.) Costly Links and (3.) Temporal Lacking. In our literature review, we already have noticed that the contemporary systems are not able address those challenges. The below we explain them in details, respectively.

(1.) **easy-to-use information access system for general purpose users** – here we outline the first challenge.

The literature review shows that systems like GoRelations [40], KOIOS [13], Hermes [104] etc. provide easy-to-use query technique. However, we understand that those systems still require some know-how of query building. For example in GoRelations, users need to know relations and concepts of the query, and then follow special techniques to construct a Linked Data query. On the other hand, in QUICK [117], Treo [35] users need to guide the query. Likewise, guidance is also required in AutoSPARQL [62], where feedback of users are used by machine learning tool to generate the SPARQL query. Therefore, the above mentioned systems are still not easy for general-purpose users who do not have enough know-how of Linked Data and its associated technologies. Even knowing of dataset vocabularies does not suffice to retrieve the required information, which impose burden to users to learn the Linked Data structure, query development, feedback giving etc.

Therefore, because of users' familiarity and comfortability towards keyword-based or natural language-based information accessed systems [77, 91], keyword-based query should be easy to use. In Linked Data information access, the keywords are made from the natural language input query, and they are posed according to the word-order of input query. The keywords directly relate label information of data. Usually users can learn such labels by observing Linked Dataset. Or, data labels can be extracted using state-of-the art entity linking [38].

(2.) **effective information access framework that hides Linked Data information access complexity** – here we outline the second two challenge.

The literature review shows that most of the contemporary systems PowerAqua [66], QAKiS [20], SemSek [4], TBSL [107] etc. use keyword-based or natural language-based information access over Linked Data because of their easy usage [77, 91]. On the other hand, those systems need to find a way so that they can follow Linked Data links. But mere keywords does not contain required semantics that is necessary to follow the links. Therefore, we see that information access systems use language tools to extract some semantics, which could be used to find the links. However, performance of those systems rely upon the performance of those language tools. Moreover, those systems can not provide a defined guide-line that what should be effective way to use those language tools.

For example, we find that the contemporary systems use language tools to generate Linked Data templates [95, 107], which later are converted to SPARQL queries. Usually the Linked Data templates provide a specific pattern or link following path at the time of query, therefore, they are potential to address the second issue that we stated above. Moreover, templates also can incorporate data semantics. However, the wrong generation of template produces incorrect answers or empty answers, which is frequently seen over the contemporary systems. This is because, we see that the template generation technique of those systems is not stable and fully defined. The correct template finding is still not solved, therefore together the two issues are still open research issue. The below we outline the template generation idea in contemporary systems, which will explain why they fail.

Usually contemporary systems generate templates for keywords. Here, the template construction procedure is: first the input query is parsed by language tools such as language parsers. Then, the parsed outputs, which generally are parse trees, are mapped to Linked Data resource which follow the Linked Data's links. A parse tree is a tree with tagged[5] part of input query. But template constructed by the language tool-based systems are not stable, therefore the templates are also not defined. In most of the cases, language tools generate multiple parse trees [107]. Or, they sometime tag input keywords with wrong tags. Or, they sometime map wrong Linked Data resources. For example, consider an input query "Japanese national whose spouse born in Germany" which holds a concept "Japanese national", two predicates "spouse" and "born", and an individual entity "Germany". Over Linked Data graph, usually a concept represents a destination node or object in triple while an individual entity represents either a source node or a destination node[6] i.e., subject or object in triple. On the other hand, a predicate always represents an edge. To fit the above input query in Linked Data graph, parse tree need to tag input query correctly. However, language tools sometime tag input query wrongly (e.g., whether "spouse" is a predicate or concept). Furthermore, even if language tool can tag input query correctly, because of multiple parse trees, correct parse tree choosing could be wrong which leads improper prediction of Linked Data subgraph, which leads wrong template generation. The wrong templates generate improper ranking. As its consequence, we access wrong information for the input query. We understand that templates can be effective in Linked Data information access, however contemporary systems do not outline a proper strategy that how templates can be generated, ranked and merged. On the other hand, selection of all

[5]tagging could be POS tagging, NER tagging etc.
[6]http://www.linkeddatatools.com/introducing-rdf

such templates are not be an efficient technique for retrieval of data from a large dataset [95].

Therefore, we should propose a guided framework on template construction, ranking, and adaptation for use with keyword-based Linked Data access. The template management should be done automatically such as by observing the dataset statistics. It would increase the user adaptability for the general-purpose Linked Data users. Thereby, we can address the second challenges that we stated above.

(3.) **temporal feature embedding in Linked Data query** – here we outline the third challenge.

We find that though extraction of temporal feature related information of typical document-based data has quite a long history of research, the same kind of research for Linked Data has been comparatively less pursued [85]. So, the temporal feature related information extraction is still not solved yet.

The literature review shows that the temporal feature related information extraction of Linked Data is relatively new. Among the few prior initiatives, system propose by Vandenbussche et al. focus on image extraction, but not focused for general perspective ([109]). Khrouf et al. and Troncy et al. both show *event* related ontology integration, i.e., what (event), when (date/time), where (geo) and who (participant of an event). But these studies consider that Linked Data would be presented with a fixed ontology (namely, LODE) and they ignore the possibility of accessing other temporal ontologies ([54, 105]), so they are also not focus for general perspective temporal. Therefore, as the general-purpose information access system, the contemporary information access systems can not embed temporal feature in their queries. It informs that the third challenge is still an open research issue.

Therefore, we should propose an easy-to-use keyword-based Linked Data information access framework that is not susceptible to any specific ontology, and can capture rich temporal semantics.

2.2.3 Our Proposal

On information access over Linked Data we propose two frameworks BoTLRet [83, 84, 86], TLDRet [85]. Both of them are keyword-based information access frameworks which are easy and effective to use by the general-purpose users.

(1.) **easy-to-use information access system for general purpose users**

- **proposal in brief** – to address the first challenge shown in Section 2.2.2, we propose a keyword-based Linked Data information access framework. Here the keywords are made out of natural language query, and they are posed according to the word-order of the query. The keywords link datasets' vocabularies, which are data labels. Since keyword-based information access is familiar and comfortable information access option, we consider that it increases user acceptability towards the proposed framework.
- **method in brief** – in the proposed framework we use keywords to construct templates. For each query question, our intuitive initial assumption is that the input keywords will be presented in order, and that by using that order there is a strong possibility of ascertaining the relationship between two adjacent keywords. Michael A. Covington supported this assumption in his work [29] where he showed that major languages such as Chinese, English, and French almost never allow variations in the word orders that make up sentences. In this perspective, our proposal is sensitive in order of keywords given in the query. Therefore, languages that are less sensitive in word-order e.g., Bangla, Japanese etc. might be difficult to adapt in the proposed framework. However, we consider that users can still retrieve their required information just by adjusting the word-order of the query.

  The proposed framework is **BoTLRet** . We describe it in Chapter 3.

(2.) **effective information access framework that hides Linked Data information access complexity**

- **proposal in brief** – to address the second challenge shown in Section 2.2.2, we propose a guided framework on template construction, ranking, and adaptation for use with keyword-based Linked Data access. The template management are done automatically by observing the dataset statistics.
- **method in brief** – in the proposed framework we use keywords to construct templates. We adapt a greedy template construction technique

which automatically conforms Linked Data's data structure, therefore they do not suffer problem that language parser based template construction suffer. Later on, the templates are adapted to their equivalent SPARQL queries to access required information.

We construct template by considering basic data structure of Linked Data. We also introduce novel template merging approach when there are several templates. The merged template are used to construct the final SPARQL query that holistically represent the entire query.

In template ranking, we introduce dataset's internal statistics checking which efficiently can predict possible templates among such many. We begin by assuming that Linked Data includes a structured representation of data in itself, which means that internal statistics such as the positional frequency of a resource can provide significant indications for use in template construction and ranking. This assumption was motivated by the query likelihood model [70], which is a traditional information retrieval (IR) model that contends that the potential for successful data retrieval is proportional to the frequency at which the required information appears in the underlying data.

The proposed framework is **BoTLRet** . We describe it in Chapter 3.

(3.) **temporal feature embedding in Linked Data query**

- **proposal in brief** – to address the third challenge shown in Section 2.2.2, we propose an extension of previous framework BoTLRet that embeds temporal feature in template, and filters out information that do not conform temporal semantics.
- **method in brief** – in the proposed framework we show that how temporal information is presented in the query. We propose how some query keywords can be considered as *temporal keywords* – keywords with time or events. We also propose how the temporal keywords can be classified into explicit temporal keywords, and implicit temporal keywords, which retrieve Linked Data information for complex temporal semantics. The explicit temporal keywords hold explicit date and time while the implicit temporal keywords implementing date time.

  Finally we propose a methodology which describe how the normal keywords and temporal keywords can be used together to retrieve temporal feature related Linked Data information.

The proposed framework is **TLDRet** . We describe it in Chapter 4.

2.3 Assessment of Linked Data Quality

Data quality of Linked Dada heavily influences Linked Data success [8], therefore, we also investigate Linked Data for their quality. Here we will briefly outline about quality assessment, its contemporary systems and current challenges and their possible solutions.

In Chapter 1, we already discussed that Linked Data are potential to contain errors [58, 87]. The Linked Data quality assessment can be done with outlier detection. Over any dataset, an outlier is deviated data which is skewed so much from the other data which might arise suspicions that the data were generated by a different mechanism than the usual [43]. Therefore to decide outlier statistically, we can consider that normal data generation follows a "generating mechanism" e.g., some given statistical process while abnormal data deviate from this generating mechanism. This phenomenon can be used to detect outlier over Linked Data. For Linked Data, we can assume that outlier data will not follow the generating mechanism for same type of Linked Data.

So, error detection requires a similarity/distance measurement defined between/among the data [115]. The below we outline how the related works on Linked Data quality assessment serve this problem. The literature review will reveal the exact challenge that the contemporary Linked Data quality assessment systems still suffer, as unsolved issues. Later, we briefly outline our proposed solutions to solve the unsolved issues that we reveal.

2.3.1 Related Works

To address the challenge of Linked Data information access (stated in Chapter 1 Section 1.2.2) we review the contemporary error detection techniques that deal with Linked Data quality assessment. With our literature review, we categorize them into four categories: (1.) Manual Intervention-based Error Detection, (2.) Particular-data-type-based Error Detection (3.) Similar Data Source-based Error Detection, and (4.) Ontology Enrichment-based Error Detection.

In Manual Intervention-based Error Detection, human assessors are engaged to assess the data quality. We review them to address the automatic error identification task comparison. On the other hand, the Particular-data-type-based Error Detection frameworks only focus towards a specific type of errors such as error in numerical data etc. We review them to address the error type irrespective errors finding. In Similar Data Source-based Error Detection, two or more data sources are required to identify the possible errors. On the other hand, the Ontology Enrichment-based Error Detection frameworks enhanced data quality. We review them to help our methodology. The below we discuss them more details.

(1.) **Manual Intervention-based Error Detection** – This kind of studies look into the Linked Dataset and then manually devise some rules to identify the errors [1, 59, 116]. Although the studies generate decent outcomes, they require domain-level expertise. However, finding of domain-level experts is not easy. Moreover, when such experts are found, the process is still costly. Therefore, the manual-intervention based error detection studies are not easy to adapt for diverse datasets and impractical for large datasets.

The below we briefly discuss two of them.

- *CLDQA* [1] is "Crowdsourcing Linked Data Quality Assessment". In this research, authors engaged crowdsourcing as a means to handle Linked Data quality assessment. They considered that this type of assessment is challenging to be solved automatically. They analyzed the most common errors encountered in Linked Data sources. The errors were classified into three categories: "Object Incorrect/Incompletely Extracted" – incorrect object value holding triples, "Data Type Incorrectly Extracted" – incorrect instance type holding resources, and "Falsely Interlinked" – incorrect or non-existing links holding triples. Based on the classification, authors distributed the tasks into two groups of assessors: (i) a contest targeting an expert crowd of researchers and Linked Data enthusiasts; complemented by (ii) paid microtasks published on AmazonMechanical Turk. Authors evaluated that crowdsourcing-enabled quality assessment is a promising and affordable way to enhance the quality of Linked Data.
- *TELDQ* [59] is "Test-driven Evaluation of Linked Data Quality". In this research, authors investigate a dataset, and creates some patterns that identify possible erroneous entries. In their work we need to know some

form of vocabularies, ontologies and knowledge bases, which they argue that it helps to ensure a basic level of quality. Their methodology is inspired by test-driven software development. That is, the methodology tries to guess "bad smells" of data and assess the data quality. The patterns are formed based on the "bad smells" which are some SPARQL query templates. Templates can be instantiated for different pattern according to the requirement.

(2.) **Particular-data-type-based Error Detection** – This kind of studies only check for error detection for particular data-type Linked Data [34, 111]. Such as, Wienand et al. investigated to find error for numerical data-type Linked Data [111]. However, the particular-data-type based erroneous data findings ignore large amounts of (other-data-type) erroneous data.

The below we briefly discuss one of them.

- *DINDD* [111] is "Detecting Incorrect Numerical Data in DBpedia". In this research, authors proposed a methods that can find numerical type of incorporate data. They applied unsupervised numerical outlier detection technique. Their outlier detection was done by applying different methods such as Interquantile Range (IQR) [108], Kernel Density Estimation [78], and various dispersion estimators, combined with different semantic grouping methods.

(3.) **Similar Data Source-based Error Detection** This kind of studies try to find two or more data sources for same Linked Data resource and then compare the data to identify the error [19, 58, 74]. However, the similar data sources are not readily available for all kind of Linked Data resources. Moreover, when such data sources are available, cross checking of them is not easy. Therefore, the similar-data-source based error detection studies face adaptation difficulties.

The below we describe one of them.

- *LCRSCWDF* [19] is "Learning Conflict Resolution Strategies for Cross-Language Wikipedia Data Fusion". In this research, authors analyzed DBpedia dataset, and resolved conflicts for data values. They considered DBpedia 3.9 hold 2.46 billion facts, among which conflicting values are existed. Therefore, they used DBpedia entries from 119 different languages, and resolved a conflicting values by applying an algorithm that considers values written in different language versions. The resolution algorithm, which is customizable for each relation, is given with some ground truth, so that by analyzing the ground through, algorithm can decide which fact should be more correct than others.

(4.) **Ontology Enrichment-based Error Detection** This kind of studies also need to check the data manually [61, 64, 79, 102]. In some cases, ontology-enrichment can be done automatically such as the work done by Lehmann et al [79] . In this research, authors automatically typified Linked Data resources that do not hold type information; however their research focus was not for error detection. Our proposed framework can be adapted on the top of their proposal because we utilize type (i.e., Class) information as the input.

- *TINRD* [79] is "Type Inference on Noisy RDF Data". This research does not fully focuses for error detection or quality assessment, rather it provides an ontology enhancement technique. However, we can consider the enhancement requirement is only necessary when the data are not completely defined. In that sense, TINRD can identify incomplete data, therefore we, to some extent, can consider it as a quality assessment framework. Anyway, in this research authors propose a statistical data type identification. To identify the data type, for each relation, authors checked incoming (subject) and outgoing (object) links or URIs. Then considering those links (subjects and objects) and their used types (types of subject and types of objects), it calculates a probability of the type. Then, if any link (subject or object) that does not hold type information, but get linked with the relation it can typify the link with the probabilistic type.

2.3.2 Unsolved Issue

To assess Linked Data quality, in Section 1.2.2, we addressed the problem – Wrong Entries in existing Linked Dataset. The below we explain the problem by analyzing the literature review.

(1.) **automatic data quality assessment framework for Linked Data, irrespective of their data types** – here we explain the above stated research problem.

In our literature review, currently we do not find any Linked Data quality assessment framework that can automatically assess the data for any error possibilities. For example, work done in [1, 59, 116] is manual-intervention based error detection frameworks. Although they generate decent outcomes, they require domain-level expertise. Furthermore, work done in [19, 58, 74] is similar data source-based error detection which require to find two or more data sources for same kind of data. However, the similar data sources are not readily available for all kind of Linked Data, or checking of them is not easy. Also, we understand that error detection framework should not only focus some particular error types [34, 111]. So, the contemporary researches can not assess Linked Dataset for wide-rage error possibilities. Therefore, we understand that identifying Linked Data errors for various type of errors is a open research issue.

Therefore, we should propose an error detection framework that can automatically assess Linked Dataset for any error possibilities, without requiring any same kind of dataset.

2.3.3 Our Proposal

On Linked Data quality assessment we propose framework ALDErrD [87].

(1.) **automatic Linked Data error assessment**

- **proposal in brief** – to address the unsolved issue, we propose an error detection framework that can automatically assess Linked Dataset for any error possibilities, without requiring any same kind of dataset.
- **method in brief** – in the proposed framework we consider that the same type of Linked Data should share the same kind of values. For example, height of a "Basketball Player" (i.e., type of data) should follow similar kind of values like other "Basketball Players". Such as, we expect individuals who are Basketball Players to be taller.

  The erroneous data values of Linked are measured by data outlier.

2.4 Summary

In this Chapter we discuss background of our thesis. We understand that Linked Data currently contain huge amount of knowledge which makes this data worthy to investigate. Therefore, at first we briefly introduce the Linked Data technologies. Then we discuss on two Linked Data issues – information access over Linked Data and data quality assessment for Linked Data. We understand that they are crucial for Linked Data success [8]. We found that although there are ample amount of researches have been conducted over these two, we find that they are still not easy and efficient.

Regarding the Linked Data information access, we understand that since Linked Data hold different data model, they are not suitable to retrieve by traditional document-based information access systems. The same is true for traditional graph search case. This is because Linked Data hold rich semantics inside the data, therefore the contemporary graph search algorithms also not suitable for data access. In our literature review we found that the contemporary systems used template-based retrieval because this technique can be adaptable with user familiar information access techniques. Furthermore, templates can be used to follow the data links easily. However, the template-based Linked Data information access studies have yet to provide concrete guidelines for template construction, ranking, and merging, all of which are required for effective adaptation of templates to Linked Data retrieval. Moreover, the contemporary Linked Data information access framework put less efforts to retrieve specific kind of information that are temporal attached. However, such information hold significant interest.

On the other hand, we find that Linked Data also hold erroneous data, which require error identification framework. Obviously manual checking of error will be best technique to judge for errors, but it is not feasible. Therefore an automatic error assessment or detection framework is necessary. However, the contemporary systems are either Linked Dataset centric, or particular data type error centric. Therefore, we should investigate a framework that will automatic, will not require Linked Dataset wise expertise, and suitable for any type of errors.

In following three Chapter we will describe their solutions.

Chapter 3

BoTRet: The Basic Information Access Framework

In this chapter, we introduce our basic information access framework. We first describe about its introduction (Section 3.1). Then we start describing our proposal (Section 3.2). At the beginning, framework can handle the query with two keywords. Later, we extend the framework for query questions with more than two keywords (Section 3.3). Thereby, the proposed framework can work for more number of keywords and retrieve information holistically that a Linked Data information access system requires. We show results of implementing our proposal through experimental results and discussion (Section 3.4). Later, we discuss on our contribution and explain the proposed framework that how it solved the problem that existed in the contemporary systems (Section 3.5) Finally, in Section 3.6 we summarize the Chapter.

3.1 Introduction

Keyword-based Linked Data information access is an easy choice for general-purpose users, but the implementation of such an approach is a challenge because mere keywords do not hold semantic information. Some studies have incorporated templates in an effort to bridge this gap, but most such approaches have proven ineffective because of inefficient template management. Usually Linked Data are presented in structured format, information access over Linked Data requires to know the data structure, their vocabularies, data ontology, SPARQL queries etc. while template can be adapted to capture those complex technologies. Furthermore, we can assume that the data's internal statistics can be used to effectively influence template management. In this work, we explore the use of this influence for template creation, ranking, and scaling. Then, we demonstrate how our proposal for automatic Linked Data information access can be used alongside familiar keyword-based information access methods, and can also be incorporated alongside other techniques, such as ontology inclusion and sophisticated matching, in order to achieve increased levels of performance.

3.1.1 Motivation

The Linked Data [10] initiative, where data are connected in a network-like structure [14] and which was motivated by the potential for link construction and identification among various data, has opened new worlds in data usage. The concept of this storage paradigm deviates from traditional repository-centric infrastructures to an open publishing model that allows other applications to access and interpret stored data [14]. As of September 2011, 295 knowledge-bases consisting of over 31 billion resource document framework (RDF) *triples* on various domains, have become interlinked via approximately 504 million RDF links [1], which got increases more than three folds just within the next two and half years, as of April 2014 [2]. Therefore, we can assume that the Linked Data are growing rapidly which contain vast amounts of knowledge.

Efficient and easy-to-use information access over Linked Data is now a necessity because these days such Linked Data hold vast amounts of knowledge. Usually,

[1]http://www4.wiwiss.fu-berlin.de/lodcloud/state/

[2]http://linkeddatacatalog.dws.informatik.uni-mannheim.de/state/

obtaining information access over a Linked Data network requires following links [5, 6, 42]. However, simply following links introduces a very basic problem, which is that the use of a network presentation makes it very hard to find endpoints, at least within a reasonable cost [2]. As a result, finding links on Linked Data is often difficult, especially for general-purpose users who have very little knowledge about the internal structure of Linked Data, such as schema information or structured query language (SQL) type expressive queries (RQL, RDQL, or SPARQL [81]).

On the other hand, the Linked Data are considered as Semantic Web data. According to the Semantic Web vision, such data should be accessible by their data semantics [11]. It advocates that Linked Data should be accessible by natural language like query, or keyword-based query. This is because, the above types of queries are intuitive to the users. Here the keywords are word segments that represent important sense of the sentence or query. In linguistics, such important word segments are mentioned as Content Words (CWs), and usually belong to nouns, most verbs, adjectives, and adverbs and refer to some object, action, or characteristic [112]. Moreover, the keyword-based Linked Data access methods are considered easy to use because of their familiarity towards the users [77, 91]. In this case, the keywords are made out of natural language query, and they are posed according to the word-order of query. Here the keywords contain one or more segments of text with their word boundaries.

To retrieve information over Linked Data, it needs to perform three tasks

(i.) decide keywords i.e., identification of important sense or part of (natural language) query i.e., CWs

(ii.) link keywords to datasets' vocabularies, which are data labels of dataset

(iii.) fit keywords to some structures so that they can retrieve the information

In this thesis, we contribute for tasks (i.) and (iii.) and keep task (ii.) as out-of-scope, which can be achieved by the state-of-the-art entity linking facilitates such as [38]. While all three tasks are equally important, we find bigger scope to investigate on task (iii.) because contemporary systems mostly suffer on it. Therefore, we dedicate Chapter 3 and 4 to describe task (iii.), and Chapter 5 to describe task (i.).

Usually, keyword word-based data access options are different from other traditional keyword-based data access types because they require adapting keywords to semantics. Since keywords do not contain semantic information (specifically ontology information), a number of researchers have proposed automatic ontology inclusion [104] to bridge that gap. However, automatic ontology inclusion is a challenge because, in such cases, the system itself needs to incorporate ontology that is, as yet, unavailable.

In attempts to resolve the above-mentioned problems, such as link finding and keyword semantics inclusion, recently, a number of other researchers have worked to incorporate templates into Linked Data keyword search schemes [95, 107]. The intuition behind "template" creation is that it would introduce a defined structure that supports finding links and their endpoints, as well as templates that could fill the semantic gaps that result when keywords alone are used. However, current template-based Linked Data access studies have yet to provide concrete guidelines for template construction, ranking, and merging, all of which are required for effective adaptation of templates to Linked Data access. In this work, we will propose a guided framework on template construction, ranking, and adaptation for use with keyword-based Linked Data access that uses internal data statistics for template management.

Our template management technique relies on keyword order. In it, templates are constructed for "adjacent keywords". More specifically, for each keyword, any other keywords that are ordered before and after are considered to be adjacent keywords. For keyword-based information access, our primary assumption is that users will enter keywords in the word order of a natural language sentence. That is, instead of inputting keywords randomly, they will input them in an order that conforms to the natural language structure of a sentence. Michael A. Covington supported this assumption in his work [29] where he showed that major languages such as Chinese, English, and French almost never allow variations in the word orders that make up sentences. In this perspective, our proposal is sensitive in order of keywords given in the query. Therefore, languages that are less sensitive in word-order e.g., Bangla, Japanese etc. might be difficult to adapt in the proposed framework. However, we consider that users can still retrieve their required information just by adjusting the word-order of the query. A word order-based information access approach is also common in contemporary semantic search research. For example, Unger *et al.* and Yahya *et al.* used a language parser to determine the word order of their natural language questions [107, 113].

3.1.2 Contributions

Among the three different tasks for a Linked Data information access framework (described in Section 3.1.1), this Chapter contributes for task (iii.). That is, fitting of keywords to some structure so that they can retrieve required information need. Therefore, the keywords that we use are already decided out of natural language query and are linked for dataset vocabularies. For example, for a natural language query "Who is Barack Obama married to?", the important parts of query are {Barack Obama, married} – that is task (i.). Here the important part of the query is "Barack Obama" and "married" and other parts "Who" "is" and "to" are not important because they are frequently used words. Therefore, "Barack Obama" and "married" intuitively mean the query. On the other hand, considering information access over DBpedia dataset, we link {Barack Obama, married} towards {Barack Obama, spouse} because they link DBpedia dataset vocabularies – that is task (ii.). In linking the keyword "Barack Obama" remains exactly same as it is in the query, while the word "married" links vocabulary "spouse" since marriage information is labeled by the word "spouse" in the dataset vocabularies. Therefore for this exemplary query, task (iii.) starts for keywords {Barack Obama, spouse}, where it fits keywords to some structures or templates to retrieve the required information need.

Over a keyword-based Linked Data information access framework perspective, exact contributions of this Chapter are the followings:

(1.) by considering Linked Data's data structure and internal statistics, we devise templates into which we can embed Linked Data's missing semantics (describe in Section 3.2)

(2.) by observing the Linked Data's internal statistics, we propose a template ranking strategy that can be used to identify a strong potential template (from among other templates) and that can be used to construct a possible SPARQL query (describe in Section 3.2)

(3.) by introducing a template merging strategy, we guide the scalability of our proposed access framework for any number of keywords (describe in Section 3.3)

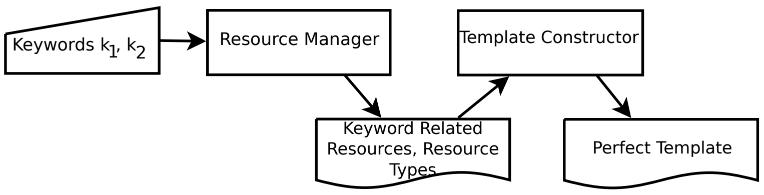


FIGURE 3.1: Template selection process flow for two-keyword query questions

(4.) by developing and evaluating the system (described in Section 3.4)

Our proposed framework **B**inary Pr**o**gressive **T**emplate Paradigm over **L**inked Data **Ret**rieval or **BoTLRet** [84, 84, 86] can automatically create templates from keywords without considering schema or ontology information through the use of an automatic Linked Data access approach, and we further hypothesize that the inclusion of such information will effectively improve the system's performance.

3.2 Two-keyword-based Linked Data Information Access

The proposed framework BoTLRet works for keywords and this section primarily focuses on how we construct a template using two adjacent keywords, and then identify the best template from among many such templates. In normal use, templates are predefined structures that are used to perform tasks by setting position holders to specific task parameters. From a Linked Data perspective, such position holders provide a predefined structure in the form of ontology (or semantics) that can be used to identify links and locate endpoints. We describe our approach as a *binary progressive approach*, which means that queries are constructed with resources from two adjacent keywords, and can then be extended to use more than two keywords. Template management for queries with more than two keywords is described in Section 3.3.

Figure 3.1 shows the two-keyword-based template selection process flow. It takes two keywords k_1 and k_2. Then, in the next step, the *resource manager* process collects and manages resources that are related to those keywords. Next, the *template constructor* process constructs a number of templates and identifies the best among them. Each of these processes will be described in greater detail below.

3.2.1 Resource Manager

The resource manager process collects and manages resources that are related to the keywords. It starts by taking two adjacent keywords and then produces *keyword-related resources* with their classifications. In the first step, for each keyword k, it extracts the keyword-related resources ($RR(k)$) from the underlying knowledge-base (KB). For Linked Data, the KB is constructed from the set of resource description framework (RDF) triples that store data using the form $< s, p, o >$ where s, p, and o are respectively considered as the *subject*, *predicate*, and *object* component elements in a RDF triple.

In the second step, for each keyword-related resource (r), the resource manager calculates three *positional frequencies* ($PF_s(r), PF_p(r)$, and $PF_o(r)$). Then, in the third step, resource manager selects a resource-type ($rType(r)$). The resource-type provides the basis of the template construction while the positional frequencies of the resource guides the process that leads to the identification of the best template. We will discuss the steps in detail below.

1. Keyword-related resource ($RR(k)$) for keyword k is a set of Linked Data resources that represent keyword k.

   $$\begin{aligned} RR(k) = \ & \{r \mid \exists < r, p, o > \in KB \land p \in rtag \\ & \land (m(o, k) \geq \alpha)\} \end{aligned}$$

   where $rtag$ is a set of resource-representing tags such as label, name title, prefLabel, etc. and $m(o, k)$ is a string-similarity function used to select the resource r that corresponds the keyword k. String-similarity is calculated between the object o of the RDF triple $< r, p, o >$ and the keyword k. r is selected for a particular similarity-threshold value α.

2. The positional frequencies ($PF_s(r), PF_p(r)$ and $PF_o(r)$) for the resource r are three different frequencies of r in the KB. Since r can be any of the three *subject, predicate* and *object* component elements in a RDF triple, subject, predicate, and object positional frequencies for r are respectively defined as follows:

$$
\begin{aligned}
PF_s(r) &= \mid \{< r, p, o > \mid \exists < r, p, o > \in KB\} \mid \\
PF_p(r) &= \mid \{< s, r, o > \mid \exists < s, r, o > \in KB\} \mid \\
PF_o(r) &= \mid \{< s, p, r > \mid \exists < s, p, r > \in KB\} \mid
\end{aligned}
$$

3. The resource-type ($rType(r)$) for resource r is a classification that classifies r as either a predicate-type (PR) or non-predicate-type (NP) resource.

$$
rType(r) = \begin{cases} PR & \text{iff}(PF_p(r) > PF_s(r)) \\ & \wedge(PF_p(r) > PF_o(r)) \\ NP & \text{otherwise} \end{cases}
$$

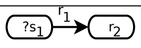
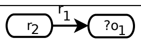
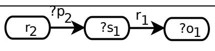
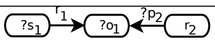
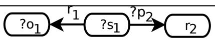
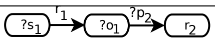
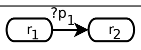
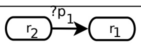
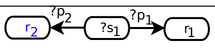
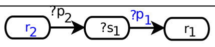
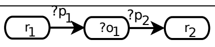
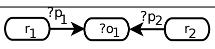
3.2.2 Template Constructor

The template constructor process constructs templates and identifies the best template. It constructs templates for two adjacent keyword-related resources on the basis of their resource-types and then identifies the most suitable template from among all those constructed.

Before defining a template, we will introduce the term “triple-pattern” ($tp(r_1, r_2)$), which is a pattern constructed for two keyword-related resources. The third column of Table 3.1 contains triple-patterns. A triple pattern picks triples from the KB. In a triple pattern, a variable resource, which starts with a question mark (i.e., ?), matches any resource in the KB and selects the matched triples. Template ($tmp(r_1, r_2)$) is set of triple-patterns i.e., $tmp(r_1, r_2) = \{tp_1(r_1, r_2), tp_2(r_1, r_2), ...\}$.

The fourth column of Table 3.1 shows a graphical illustration of each triple-pattern. A template accumulates a set of triple-patterns. Triple-patterns are constructed by considering resource types of participating resources. The second column of Table 3.1 shows such consideration by the name “condition”. Here, it can be seen that templates are constructed by maintaining two set of conditions i.e., i) rType(r_1) = PR $\wedge$ rType(r_2) = NP, and ii) rType(r_1) = NP $\wedge$ rType(r_2) = NP. However, for other two possible conditions sets, such as, for rType(r_1) = NP $\wedge$ rType(r_2) = PR, we construct templates by swapping r_1 and r_2, and for rType(r_1) = PR $\wedge$ rType(r_2) = PR, we use *modified templates* (described in Section 3.3.3). The fifth column of Table 3.1 shows *closeness* of a triple-pattern which indicates how closely r_1 and r_2 are attached.

TABLE 3.1: Templates for $r_1 \in RR(k_1)$ and $r_2 \in RR(k_2)$

	Condition	Triple pattern $tp(r_1, r_2)$	Graphical Illustration	Closeness of Triple-pattern
$tmp(r_1, r_2)$	rType(r_1) = PR $\wedge$ rType(r_2) = NP	$<?s_1, r_1, r_2>$		1
		$<r_2, r_1, ?o_1>$		1
		$<?s_1, r_1, ?o_1>< r_2, ?p_2, ?s_1>$		2
		$<?s_1, r_1, ?o_1>< r_2, ?p_2, ?o_1>$		2
		$<?s_1, r_1, ?o_1><?s_1, ?p_2, r_2>$		2
		$<?s_1, r_1, ?o_1><?o_1, ?p_2, r_2>$		2
$tmp(r_1, r_2)$	rType(r_1) = NP $\wedge$ rType(r_2) = NP	$<r_1, ?p_1, r_2>$		1
		$<r_2, ?p_1, r_1>$		1
		$<?s_1, ?p_1, r_1><?s_1, ?p_2, r_2>$		2
		$<?s_1, ?p_1, r_1>< r_2, ?p_2, ?s_1>$		2
		$<r_1, ?p_1, ?o_1><?o_1, ?p_2, r_2>$		2
		$<r_1, ?p_1, ?o_1>< r_2, ?p_2, ?o_1>$		2

Since a template (resp. triple-pattern) is constructed according to the structure of an RDF triple, it is assumed that the template incorporates Linked Data semantics and can able to retrieve the required information. For example, for keywords k_1 = spouse and k_2 = Barack Obama, it is assumed $r_1 \in RR(spouse)$ and $r_2 \in RR$(Barack Obama), and $rType(r_1)$ = PR and $rType(r_2)$ = NP, which fits into the triple-pattern $<?s_1, r_1, r_2>$ and retrieve the required information.

Each template holds multiple triple-patterns and each keyword generates multiple keyword-related resources. Therefore, to determine the best template (which we call a *perfect template* ($pefTmp(k_1, k_2)$)) for each two adjacent keywords k_1 and k_2, we need to identify the best triple-patterns among the templates that can incorporate the largest amount of the Linked Data semantics. By calculating the *relatedness* of each triple-pattern towards the KB, we can then pick the best triple-pattern and select it as the perfect template.

3.2.2.1 Selection criterion of triple-pattern relatedness

Resource frequency plays a prime role in measuring triple-pattern relatedness. We are motivated by this frequency-based approach from the classical *term frequency* (TF) calculation. From a document-based information access perspective, the TF of a term measures its importance over a particular document. From a Linked

Data information access perspective, we replace terms with keyword-related resources. Therefore, to conform a TF-like triple-pattern relatedness calculation, we hypothesize that the more frequently keyword-related resources appear in an underlying dataset, the greater its potential for use in access [70].

In a triple-pattern relatedness calculation, we consider the following: i) how similar triple-pattern's keyword-related resources are in representing the keywords, ii) how frequent a triple-pattern is, and iii) how frequently the triple-pattern's keyword-related resources appear in the triple-patterns.

Therefore, the *relatedness* value of each triple-pattern towards the KB is calculated using following statistics:

1. **String-similarity value $m(o, k)$ between the resource r and the given keyword k**: this r is the subject component element of $< r, p, o >$ where $p \in rtag$.
2. **Frequency of triple-pattern $fqTP(tp(r_1, r_2))$**: this counts the number of RDF triples associated with the $tp(r_1, r_2)$.
3. **Frequency of resource r w.r.t. triple-pattern $fqR(r, tp(r_1, r_2))$**: this is the positional frequency of the resource r in the KB where the position (*subject, predicate* or *object*) of r is guided by $tp(r_1, r_2)$.

$$fqR(r, tp(r_1, r_2)) = \begin{cases} PF_s(r) & \text{if } r \text{ is on } \textit{subject} \\ & \text{in } tp(r_1, r_2) \\ PF_p(r) & \text{if } r \text{ is on } \textit{predicate} \\ & \text{in } tp(r_1, r_2) \\ PF_o(r) & \text{if } r \text{ is on } \textit{object} \\ & \text{in } tp(r_1, r_2) \end{cases}$$

Final relatedness $fRel(tp(r', r''))$ of triple-pattern $tp(r', r'')$ is defined as

$$\begin{aligned} fRel(tp(r', r'')) = \; & m(o', k_1) * m(o'', k_2) \\ & * fqTP(tp(r', r'')) * fqR(r', tp(r', r'')) \\ & * fqR(r'', tp(r', r'')) \end{aligned}$$

3.2.2.2 Selection of the perfect template

We select the perfect template (from the all triple-patterns of all possible templates) by considering two parameters: closeness and relatedness. We first select closeness value 1 type triple-patterns with relatedness values greater than zero, sort them by their relatedness values, and then choose the highest related triple-pattern as the perfect template. If no closeness value 1 type triple-patterns with relatedness values greater than zero can be identified, we then consider the highest relatedness valued closeness 2 type triple-pattern to be the best triple-pattern, and select it as the perfect template. In any case, if we identify several best possible triple-patterns (because of identical relatedness and closeness values), we pick one at random as the perfect template. Considering the above, it can be seen that the template selector process can provide a perfect template for two given keywords. For a two-keyword query question, this perfect template related SPARQL query is then used to identify our intended result.

3.3 More-than-two-keyword-based Linked Data Information Access

In this section, we will describe how we extend previous two-keyword-based template management to handle queries with more than two keywords and show how, we use pair of keywords and pair of templates to find the merged template. Figure 3.2 shows this merged template selection procedure. It starts with the *template constructor* generating perfect templates for each two adjacent keywords. Next, the *comparator* process takes each two adjacent perfect templates and determines whether it should retain them by designating one as the *retained template (RT)* and finding a *not retained keyword (NRK)* from the other perfect template. The *refiner* process then selects some RTs as refined-RTs and generates modified templates from the NRKs. Finally, the *merger* template merging process produces the *merged template*. We will describe each process in detail below.

Throughout our description, we will use a question *"In which films directed by Garry Marshall was Julia Roberts starring?"*, for which we devise keywords $\{k_1 =$ *Film,*

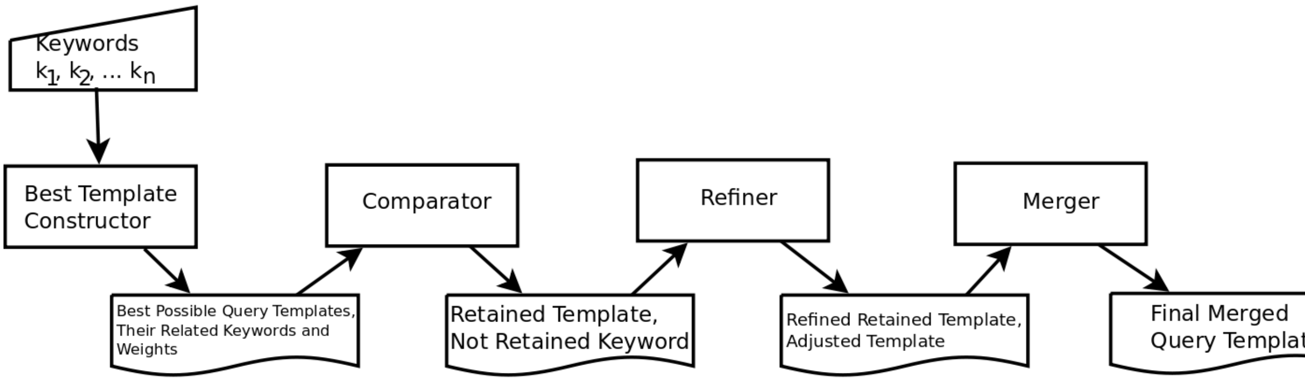

Keywords
k1, k2, ... kn
Best Template Constructor
Best Possible Query Templates, Their Related Keywords and Weights
Comparator
Retained Template, Not Retained Keyword
Refiner
Refined Retained Template, Adjusted Template
Merger
Final Merged Query Templat

FIGURE 3.2: Template selection process flow for query question with more than two keywords

k_2 =*director*, k_3 =*Garry Marshall*, k_4 =*Julia Roberts*, k_5 =*starring*$\}$ as a running exemplary question and execute our proposed framework over DBpedia[3] KB.

3.3.1 Template constructor

This process selects all perfect templates for each two adjacent keywords and stores them along with their keywords and weights. For i number of keywords, we get $i-1$ number of perfect templates for each two adjacent keywords. For example, for the running exemplary question, we get four perfect templates – the first perfect template ($pefTmp_1$) for $\{k_1, k_2\}$, the second perfect template ($pefTmp_2$) for $\{k_2, k_3\}$, and so on. In the perfect template selection, if we are unable to find any triple-pattern with relatedness value greater than zero, any triple-pattern identified will be designated as the perfect template. For the running exemplary question, we get four adjacent best possible perfect templates as follows: [4].

- $pefTmp_1$ = <$?s_1$, o:director, $?o_1$ ><$?o_1$, $?p_2$,o:Film>
- $pefTmp_2$ = <$?s_1$, o:director, r:Garry_Marshall >,
- $pefTmp_3$ = <$?s_1$, $?p_1$, r:Garry_Marshall><$?s_1$, $?p_2$, r:Julia_Roberts >,
- $pefTmp_4$ = <$?s_1$, o:starring, r:Julia_Roberts >

[3]http://dbpedia.org/

[4]o: is prefix for http://dbpedia.org/ontology/ and r: is prefix for http://dbpedia.org/resource/

3.3.2 Comparator

This process compares each two adjacent perfect templates using their closeness values and relatedness values. Using this process, we designate one perfect template as the RT and select one keyword from the other perfect template as the NRK. The RT is selected by the lower depth and higher relatedness valued perfect templates between the participating perfect templates. In contrast, NRK is found by the exclusively associated keyword held by a perfect template other than RT.

For the running exemplary question, between $pefTmp_1$ and $pefTmp_2$, $pefTmp_2$ carries lower closeness and higher relatedness values than $pefTmp_1$, we consider $pefTmp_2$ as the first RT (say rt_1) and k_1 (i.e., Film) as the first NRK (e.g., nrk_1) because k_1 is an exclusively associated keyword in $pefTmp_1$.

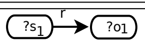
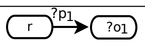
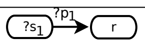
Since we follow a *binary progressive approach*, the comparator process is executed for the adjacent pairs perfect templates. Therefore, if the number of perfect templates is $i - 1$ (see Section 4.1), we compare pairs for $pefTmp_j$ and $pefTmp_{(j+1)}$, where $1 \leq j \leq i - 2$. This comparison gives all RTs and NRKs. However, in some cases, RTs and NRKs might share *common keywords*. For example, for the running exemplary question, $pefTmp_2$ associates the keyword Garry Marshall, which also appears as an NRK. Furthermore, $pefTmp_2$ appears twice as RTs which also share common keywords between them. Since we only construct templates for each keyword once, it is necessary to refine common keyword-related RTs and NRKs.

3.3.3 Refiner

This process refines previously generated RTs and NRKs so that the final template will only use each keyword once. The *refiner* process eliminate redundancies using two operations: i) eliminating redundant RTs, and ii) eliminating redundant NRKs. We will discuss the operations in details in the below:

i) As the first operation, we take a set of RTs along with their generation order, i.e., rt_1 appears first, after which the second one appears, and so on, from which we generate a set of refined-RTs. We eliminate redundancy between the RTs in order to make each RT unique. For the running exemplary question, since there are two $pefTmp_2$s RTs, we first eliminate one $pefTmp_2$ and then identify the unique RTs as $\{pefTmp_2, pefTmp_4\}$. After finding a set of unique

TABLE 3.2: Modified templates for single resource $r \in$RR(k)

	Condition	modified- $tp(r)$	Graphical Illustration	Closeness of Modified-$tp(r)$
modified-$tmp(r)$	rType(r)=PR	$<?s_1, r, ?o_1>$		1
modified-$tmp(r)$	rType(r)=NP	$< r, ?p_1, ?o1 >$		1
		$<?s_1, ?p_1, r>$		1

RTs, we determine whether the two unique RTs share common keywords and eliminate any such identified. For such cases, if there is a shared common keyword between two unique RTs, we store the most recently generated one and eliminate the other one. For the running exemplary question, the unique RTs $\{pefTmp_2, pefTmp_4\}$ do not share a common keyword, so we do not need to eliminate any of them. At the end of the first operation, the unique RTs that remain are considered to be refined-RTs. Therefore, for the running exemplary question, the refined-RTs are $\{pefTmp_2, pefTmp_4\}$.

ii) For the second operation, we take a set of NRKs and refined-RTs and generate a set of refined-NRKs. Each NRK is checked to determine whether it is already associated with any of the refined-RTs. Those that are not are considered refined-NRKs. For example, for the running exemplary question, we get {Film} as a refined-NRK as "Film" is not associated with any of the refined-RTs.

Since it is necessary to use a template for refined-NRK, we convert each refined-NRK k to its perfect template, which is called a modified perfect template (modified-perTmp(k)). Our earlier template generation was intended for two keywords, but in this step we modify template generation for a single keyword (i.e., for each refined-NRK). To accomplish this, we find keyword-related resources for each refined-NRK along with their type classifications, and then construct single-resource-based triple-patterns (modified-$tp(r)$s) and modified templates (modified-$tmp(r)$s), as can be seen in Table 3.2.

For each refined-NRK k, we identify the best modified triple-pattern as the modified perfect template from among all modified templates that possesses maximum relatedness value towards the KB. The relatedness of a modified triple-pattern is counted by considering how many RDF triples are associated in the KB.

For the running exemplary question, we get $<?s_1, ?p_1,$ o:Film$>$ as modified-pefTmp(Film). This single-keyword-based template also supports template construction when two adjacent keyword-related resources appear as predicate type resources (see Section 3.2). In such cases, for each predicate type keyword-related resource, we construct a modified triple-pattern.

As a result, the refiner process generates modified perfect templates for all refined-NRKs, and then forwards all refined-RTs and modified perfect templates to the next process.

3.3.4 Merger

This process, which produces a merged template for all query keywords, begins after the generation of all refined-RTs and modified perfect templates. Here, we merge refined-RTs and modified perfect templates according to the given keyword order. Template merging can be considered triple-pattern merging because each individual refined-RT and individual modified perfect template are nothing more than single triple-patterns. Therefore, in the following paragraphs, template merging is described as triple-pattern merging.

We merge triple-patterns by introducing *connectors*. Connectors are merging points where triple-patterns are merged with one another. We use a triple-pattern's variable resource holding subject and object component elements as connectors. For example, for a triple-pattern $<?s_1, r_1, ?o_1>$ $<r_2, ?p_2, ?s_1>$, $?s_1$ and $?o_1$ are connectors. Since each triple-pattern holds multiple connectors, we greedily try to merge them until we find a valid merged triple-pattern. The validity of the merged triple-pattern is checked by its SPARQL query, which determines whether the merged triple-pattern corresponding to the SPARQL query generates any non-empty output. To serve this greedy approach, we assign priorities to the triple-pattern connectors and then merge triple-patterns according those priorities.

3.3.4.1 Triple-pattern connector priorities

The priorities of triple-pattern connectors will vary depending on how many connectors each triple-pattern holds. Below is the calculation used to assign priority:

i) For a two-connector based triple-pattern, a connector that contributes as the RDF triple component element with a later order keyword-related resource is considered the *higher priority connector*, while the other connector is considered the *lower priority connector*. If both connectors appear in the same RDF triple component, the connector that contributes as the subject component element of the RDF triple is considered to be the higher priority connector. For example, for a triple-pattern $<?s_1, r_1, ?o_1 >< r_2, ?p_2, ?s_1 >$ which is constructed for two resources of two orderly given keywords k_1 and k_2, $?s_1$ is the higher priority connector while $?o_1$ is the lower priority connector because s_1 is associated with the later keyword k_2 (through the r_2). In contrast, for a triple-pattern like $<?s_2, r, ?o_2>$ where both connectors appear in the same RDF triple component, $?s_2$ holds a higher priority than $?o_2$.

ii) For one connector based triple-pattern, the connector is simultaneously considered both the higher priority connector and lower priority connector.

3.3.4.2 Merging of triple-patterns

In the binary progressive approach, we start triple-pattern merging for the first two triple-patterns by attempting to merge higher priority and lower priority connectors. If we can obtain a valid merged triple-pattern, we then consider this as an intermediate merged triple-pattern. Then, we merge the next triple-pattern to the connectors of the intermediate merged triple-pattern (i.e., come from its constituent triple-patterns).

Such iterative triple-pattern merging is continued until we merge the last triple-pattern. However, in intermediate merged triple-pattern generation, the connector priorities get updated during each valid merged triple-pattern finding. Below we describe the priority update process for the connectors of each intermediate merged triple-pattern.

i) Connectors that belong to the lastly merged triple-pattern hold higher priorities. If the triple-pattern merged last holds two connectors, the highest priority goes to the connector that can generate a valid merged triple-pattern followed by the second connector.

ii) The connectors that remain follow the updated priorities.

For example, for an intermediate merged triple-pattern $<?s_1, r_1, ?o_1 ><?o_1, r_2, ?o_2 >$ where $?o_1$ holds the (so far) highest priority followed by $?o_2$ and $?s_1$, and a next triple-pattern $<?s_2, r_3, r_4 >$ attempting to merge at the connector $?o_1$ but are unable to generate a valid merged triple-pattern, but do get merged at the connector $?o_2$ where they are able to generate a valid intermediate merged triple-pattern $<?s_1, r_1, ?o_1 ><?o_1, r_2, ?o_2 ><?o_2, r_3, r_4 >$, then, in the new intermediate merged triple-pattern, $?o_2$ holds the (new) highest priority followed by $?o_1$ and $?s_1$. In this manner, priority-based merging reduces merging complexity.

Therefore, for the running exemplary question, we obtain a valid merged triple-pattern (or template) as

```
<?s1,?p1, o:Film>.
<?s1 , o:director , r:Garry_Marshall>.
<?s1 , o:starring , r:Julia_Roberts >.
```

3.4 Experiment

In this experiment, we use question answering over Linked Data 1 and 2 (i.e., QALD-1[5] and QALD-2[6]) open challenge test question sets. Both challenges include the same type of natural language training and test question sets from DBpedia and MusicBrainz[7].

Since this Chapter contributes for task (iii.) (described in Section 3.1.1 and 3.1.2) i.e., the fitting of keywords to some structure so that they can retrieve required information need, the keywords that we use are already decided out of natural language query and are linked for dataset vocabularies. For example, for a natural language query “Who is Barack Obama married to?”, the input keywords for BoTLRet are {Barack Obama, spouse}. This is because words {Barack Obama, married} carry important sense of the query, and word “Barack Obama” links keyword “Barack Obama” and word “married” links “spouse” over DBpedia vocabularies. We consider the linking task i.e., the task (ii.) is “out of scope” of this thesis, therefore the exemplary query starts for keywords {Barack Obama,

[5]http://greententacle.techfak.uni-bielefeld.de/˜cunger/qald/index.php?x=home&q=1

[6]http://greententacle.techfak.uni-bielefeld.de/˜cunger/qald/index.php?x=home&q=2

[7]http://musicbrainz.org/

spouse}. Moreover, the order of the keywords is as the keywords are appeared in the question.

The DBpedia dataset used in our experiment comprises more than 30 million instances, 288 thousand classes, and almost 50 thousand properties, while the MusicBrainz dataset comprises almost 4 million instances, 31 classes, and 125 properties. A resource is a class that appears with [8]type.

The proposed framework can not handle questions that require *Boolean* type answers, aggregation functions, temporal precedence (such as *latest*, *past five years*, etc.) understanding, and questions for which resources are not found in the KB, because they are out of the scope of our research. We used 78.12% of the QALD-2 DBpedia test questions and 74% of the QALD-2 MusicBrainz test questions.

The QALD-2 test questions were used for our detailed experimental evaluation. In contrast, QALD-1 questions were used to compare other keyword-based data access initiative. In the comparison between BoTLRet and other keyword-based system, we were able to compare 66% of the QALD-1 DBpedia test questions because the comparison was performed for questions that are executed by both BoTLRet and other systems. When the other systems did not execute MusicBrainz questions, only the DBpedia dataset questions were used.

We mainly report experimental results with standard information retrieval evaluation metrics recall, precision and F 1 measure . The execution costs are shown in Seconds or Milliseconds.

To select keyword-related resources via the resource manager process, we use similarity-threshold value $\alpha = 1$. For DBpedia and MusicBrainz datasets, we manually define *rtag* as {[9]label, [10]title}. We then implement BoTLRet using the Java Jena (version 2.6.4) framework. The BoTLRet hardware specifications are as follows:

Intel®Core$^{\text{TM}}$i7-4770K central processing unit (CPU) 3.50 GHz based system with 16 GB memory. We loaded DBpedia and MusicBrainz KBs in Virtuoso (version 06.01.3127) triple-store, which was maintained in a network server. To evaluate BoTLRet, we performed three experiments and analyzed their results. These experiments will be described in detail below.

[8]http://www.w3.org/1999/02/22-rdf-syntax-ns#
[9]http://www.w3.org/2000/01/rdf-schema#
[10]http://purl.org/dc/elements/1.1/

3.4.1 Experiment 1

The first experiment was performed to evaluate how the BoTLRet performs for two-keyword-based access and more-than-two-keyword-based access. therefore, we will report BoTLRet performance according to the keyword group, i.e., number of keywords each question holds. Keyword groups are separated by the number of keywords each question can hold. For example, the exemplary question shown in Section 3.4 falls into a "two keyword group" question, because it holds five keywords.

We executed the BoTLRet system for each group of questions and evaluated their results in terms of average recall, average precision, and average F1 measure. Based on the given answers of QALD-2 test questions, recall is the fraction of relevant answers that the BoTLRet can retrieve. Precision is the fraction of retrieved answers that are relevant, and the F1 measure is the harmonic mean of precision and recall. For each of the keyword-group questions, we calculated the average precision and average recall by summing up the individual recall and individual precision, and then dividing them by the number of questions for each group. However, it was impossible to calculate the average F1 measure using the same method because the individual F1 measure cannot be calculated if the recall of that individual question is zero. In such cases, we put the individual question's F1 measure at zero as well, and then calculated the average F1 measure for each group of questions.

Additionally, for the DBpedia dataset, the average execution costs for two, three, four, and five keyword group questions were measured as 41.00, 91.74, 134.27, and 164.02 seconds respectively – which is a linearly increased trend.

Table 3.3 shows our keyword-group-wise result analysis for recall, precision, and F1 measure. The bottom of the table shows averages for the recall, precision, and F1 measure for both set of questions. As you can see, the performance of the "two keyword group" questions indicates that our template selection proposal works well.

This also ensures usefulness of triple-pattern relatedness calculation i.e., $fRel(tp(r', r''))$ (shown in Section 3.2.2.1). The performance of the questions for more than two keywords also validates our template merging policy. Therefore, we conclude that

TABLE 3.3: BoTLRet Recall, precision, and F1 measure grouped by number of keywords for the DBpedia and MusicBrainz questions

	DBpedia				MusicBrainz			
	No of Qs	Recall (avg)	Precision (avg)	F1 Measure (avg)	No of Qs	Recall (avg)	Precision (avg)	F1 Measure (avg)
2	51	0.961	0.961	0.961	7	1.000	1.000	1.000
3	13	0.923	0.852	0.857	8	1.000	1.000	1.000
4	6	0.833	0.833	0.833	16	0.875	0.875	0.875
5	5	1.000	1.000	1.000	5	0.800	0.800	0.800
6	-	-	-	-	1	1.000	1.000	1.000
	Average	0.946	0.943	**0.944**	Average	0.918	0.918	**0.918**

internal structure of the Linked Data and their statistics have more significant impact on template construction, which can be used potentially over keyword-based Linked Data information access.

3.4.2 Experiment 2

The second experiment was performed to evaluate the effectiveness of resource classification of our proposal when selecting valid triple-patterns. To accomplish this, we investigated the triple-pattern generation approach of BoTLRet with an exhaustive system (referred to hereafter as a maximum triple-pattern system (MTS)), which uses an exhaustive triple-pattern generation approach to compare the performance of BoTLRet and the performance of MTS in result generation. We then report the primacy of one system over the other.

In principle, the framework details of both BoTLRet and MTS are nearly identical. However, in the BoTLRet, we construct triple-patterns by considering the classifications of keyword-related resources, while in the MTS, we construct triple-patterns without considering the classification. Therefore, the MTS considers all possible combinations for keyword-related resources when constructing triple-patterns and can be considered to be an exhaustive version of the BoTLRet.

Next, we compared comparative computational cost requirements between the two systems. Triple-pattern usage by the BoTLRet system can be divided into three cases:

Table 3.4: Comparison between MTS and BoTLRet systems in terms of number of triple-patterns used and computational cost

case	Used No of Triple-patterns		Computational Cost by BoTLRet w.r.t. MTS
	MTS	BoTLRet	
PR-NP	49	7	0.142
NP-NP	49	8	0.163
TOT	49	15	0.306

Table 3.5: Completeness comparison between MTS and BoTLRet

	MTS			BoTLRet		
	Recall (avg)	Precision (avg)	F1 Measure (avg)	Recall (avg)	Precision (avg)	F1 Measure (avg)
DBpedia	0.959	0.956	0.957	0.946	0.943	0.944
MusicBrainz	0.918	0.918	0.918	0.918	0.918	0.918

- PR-NP: Triple-patterns that hold one predicate type keyword-related resource. Seven triple-patterns belong to this case.
- NP-NP: Triple-patterns that hold two non-predicate type keyword-related resources. Eight triple-patterns belong to this case.
- TOT: Triple-patterns that hold both PR-NP and NP-NP cases. Fifteen triple-patterns belong to this case.

Since MTS always uses 49 triple-patterns, it is clear that the BoTLRet system requires less execution time. Table 3.4 shows an efficiency comparison by dividing them into three triple-pattern cases, i.e., PR-NP, NP-NP, and TOT, between the two systems. As the number of triple-patterns used is less for BoTLRet in all the three cases, it is clear that the computational costs of our method are significantly lower than that of MTS. We then tested MTS and BoTLRet for recall, precision, and F1 measure while checking for performance deterioration. Table 3.5 shows the result of this comparison for DBpedia and MusicBrainz questions. Despite significant reductions in the required number of RDF triple searches by BoTLRet, we found that the BoTLRet performance was roughly equivalent to that of MTS.

In BoTLRet, selection of a triple-pattern case (i.e., PR-NP, NP-NP, or TOT) could be considered as an overhead because, in such a selection, BoTLRet requires positional frequencies, i.e., $PF_s(r), PF_p(r)$, and $PF_o(r)$ (shown in Section 3.2.1),

Table 3.6: Execution cost (in milliseconds) for QALD-2 DBpedia test questions

One Keyword Group	Two Keyword Group	Three Keyword Group	Four Keyword Group	Five Keyword Group
710	2441	2774	3585	3720

for each of the keyword-related resources. However, even with this overhead, the computational cost of BoTLRet is lower. For example, for a two keyword group query, if we have m and n number of keyword-related resources, BoTLRet requires a $3 * m * n$ unit overhead computational cost to decide a triple-pattern case. However, this helps BoTLRet to reduce the required number of triple-pattern constructions to either $7 * m * n$, $8 * m * n$, or $15 * m * n$, while, for the same setting, MTS always constructs a $49 * m * n$ number of triple-patterns. Furthermore, this overhead computational cost can be avoided by introducing pre-calculated positional frequencies. We showed this pre-calculation-based execution cost in a framework called **inte**lligent **search** system (inteSearch) [83]. With the pre-calculated statistics, we found that a "five keyword" holding query on an average[11] generated results within 3.72 Seconds. To show execution cost, we grouped QALD-2 questions by the number of keywords each question holds that is "keyword group", executed them by inteSearch over the pre-calculated statistics and calculated their average execution cost. We found that QALD-2 questions hold five groups of keyword questions: one to five keyword group questions. Table 3.6 shows execution cost (in milliseconds) for QALD-2 DBpedia test questions. Although the execution cost could be reduced further, we consider that current execution cost is still reasonable.

Anyway, based on the results shown in Table 3.4 and 3.5, we conclude that, even with BoTLRet's quite low computational cost, the system shows almost the same level of performance as an exhaustive system such as MTS. Therefore, it can be said that BoTLRet fulfills the completeness competency requirement considering current proposal of template constructions and their merging.

3.4.3 Experiment 3

The third experiment was performed to evaluate the performance with other systems. Firstly, we sought to compare BoTLRet with other keyword-based system.

[11]calculated for running same queries for three times

Table 3.7: Performance comparison between GoRelations and our system for QALD-1 DBpedia test questions

	Recall	Precision	F1 Measure
GoRalations[40]	0.722	0.687	0.704
BoTLRet	0.825	0.793	**0.801**

However, we were unable to find any keyword-based system had been evaluated using QALD-2, although we did find a system called GoRelations[40] which is, in some sense, a keyword-based system and which used the QALD-1 question set in its evaluation. Therefore, we compared BoTLRet and GoRalations using the QALD-1 test question set. Of the 50 questions in the set, we found that both GoRalations and BoTLRet were able to execute 33 questions in common, which were then compared for average recall, average precision, and average F1 measure. Table 3.7 shows that BoTLRet outperformed GoRelations in all three areas.

Next, we evaluated the performance comparison between BoTLRet and QALD-2 challenge participant systems, specifically SemSek, Alexandria, MHE, and QAKiS [67]. However, it is necessary to mention that the systems were not fully comparable, because BoTLRet is a keyword-based system, while the others are natural language based systems. However, we present this performance comparison based on the assumption that if the required keywords are given to BoTLRet, BoTLRet will work in a very sophisticated manner. We also acknowledge that automatic identification of such keywords will further increase complexities.

For BoTLRet, the answered questions were 75 DBpedia questions that had also been used in Experiment 1. Table 3.8 shows a performance comparison between the QALD-2 challenge participant systems and BoTLRet. In the evaluation report[12], the challenge participant systems reported on how many questions each system answered. Next, for the answered questions, each system reported its average recall, average precision, and average F1 measure. Table 3.8 columns two, three, four, and five, respectively, show these performance levels. The results are shown for DBpedia test questions because of their availability.

It can be seen that BoTLRet performs far better than the other systems. It indicates, for Linked Data information access systems, the necessity of harnessing the internal structures and statistics of Linked Data.

[12]http://greententacle.techfak.uni-bielefeld.de/~cunger/qald/index.php?x=home&q=2

Table 3.8: Performance comparison between QALD-2 challenge participant systems and BoTLRet for DBpedia test questions

	Total answered questions	Recall	Precision	F1 Measure
SemSek	80	0.48	0.44	0.46
Alexandria	25	0.46	0.43	0.45
MHE	97	0.40	0.36	0.38
QAKiS	35	0.37	0.39	0.38
BoTLRet	75	0.94	0.94	0.94

Although BoTLRet is not fully comparable with the QALD-2 challenge participant systems, because in BoTLRet we used manually devised keywords while the participant systems did not, we include this experiment to show how BoTLRet would perform, if the required keywords are given correctly. We consider that the state of the art tools such as language parser, machine learning based keyword finder, and entity linker can be used find the correct keywords. In Chapter 5, we discuss it in more details.

3.5 Discussion

Here we discuss some findings of our proposed framework BoTLRet. At the first part, we discuss on some issues that we should consider to use the framework. At the second part, we discuss how BoTLRet addresses some research challenges over Linked Data information access.

3.5.1 Consideration of Use of BoTLRet

In BoTLRet, the template management technique relies on keyword order, that is templates are constructed for "adjacent keywords". More specifically, for each keyword, any other keywords that are ordered before and after are considered to be adjacent keywords. For keyword-based information access, our primary assumption is that users will enter keywords in the word order of a natural language sentence. That is, instead of inputting keywords randomly, they will input them in an order that conforms to the natural language structure of a sentence. In this perspective, our proposal is sensitive in order of keywords given in the query.

Therefore, languages that are less sensitive in word-order e.g., Bangla, Japanese etc. might be difficult to adapt in the proposed framework.

However, as we adapt greedy template management technique where we

- select the best template according to the dataset statistics
- merge adjacent templates with their best template

It automatically adjust the best possible word order of a keyword among the three adjacent keywords. For example, for three adjacent keywords $\{k_1$, k_2 and $k_3\}$, the best templates can be constructed for $\{k_1$ and $k_2\}$ or $\{k_2$ and $k_3\}$, therefore, to some extent, the keyword order is automatically adjustable. However, if number of keywords are more than four, this automatic approach will not work. In such a case, we consider that users can still use our framework by adjusting the word-order of the query.

Furthermore, we also find an issue in keyword-based information access, which is a more general problem in such technique. While a natural language query is more clearly defined, a keyword-based query some time is vague and not clearly defined. For example, keywords {Barack Obama, child} could be meant for children of Barack Obama, or they could meant for parents of Barack Obama. Over Linked Data information access, they need to handle with different keywords. For example, to know the children we need to pose the query with {Barack Obama, child} while to know the parent we need to pose keywords {Barack Obama, parent} and so on.

3.5.2 BoTLRet Addressed Research Challenges

Over Linked Data information access, BoTLRet addresses first two research challenges that we described in Chapter 1 Section 1.2.1. They are about

- “Complex Competency” where we described that information access over Linked Data requires complex user competency, from data modeling to data queries. As a result, accessing of Linked Data is often not easy, especially for general-purpose users who have very little knowledge about the internal structure of Linked Data.

- "Costly Links" where we described that information access over Linked Data network requires following links, which turn Linked Data search as a subgraph search. However, a subgraph search over a large graph is a subgraph isomorphism problem and is a classical NP-complete problem. Moreover, the traditional graph queries are not able to capture Linked Data's rich semantics. As a result, accessing of Linked Data requires new kind of data link following that can handle exponential link following complexity but still can capture Linked Data's rich semantics.

The below we discuss about how BoTLRet address these two challenges.

- First, BoTLRet is a keyword-based systems, which addresses the first challenge. The users of our framework do not need to define extra information such as data type, or any special query technique etc. for the keywords, which usually are required for systems like [30, 31, 40]. Furthermore, users do not need to think of Linked Data's complex data structure in data query. As users are familiar and comfortable with natural language-based or keyword-based query technique [77, 91], we consider that users of our frameworks will find them easy. Thereby, we address the first challenge.

  Furthermore, although the current framework requires users to manually construct the keywords, which users can devise by learning the dataset. An automatic approach can be adapted for this, which we will describe in Chapter 5.

- Second, BoTLRet adapts template-based Linked Data information access technique, which addresses the second challenge. The contemporary systems used templates to find data links over Linked Data [95, 107]. Usually templates reduce data link finding complexity. Moreover, they also incorporate Linked Data's rich semantics. These factors instigated us to adapt the template-based information access technique. However, when templates are constructed for input keywords, the contemporary systems fail to manage them effectively because most of them used language-based tools such as language parser which are not stable, and are not well defined [30, 31, 66, 107]. As a consequence, the contemporary systems sometime construct wrong templates which eventually generate wrong or empty results. On the other hand, selection of all such possibilities are not be an efficient technique for retrieval of data over a large dataset [95].

In our proposal, we tackle the mentioned challenge with greedy template management technique. We analyze Linked Data structure and propose fifteen different templates (shown in Table 3.1 and 3.2). The proposed template generation technique is stable and well defined. According to Linked Data statistics, we can guide that in what situation which template need to generate. It solve the problem of unstable template generation that exists in the contemporary systems [30, 31, 66, 107]. Moreover, template merging technique that we proposed in Section 3.3 gives defined direction how to handle query with more than two keywords. It conforms information access requirement over Linked Data that guides that Linked Data information access needs to be done holistically (described in Section 1.2.1).

Moreover, the greedy template generation technique that we adapt in our frameworks is computationally effective. It reduces data link finding complexity. This is because, if we follow all possible template generation possibilities, we should propose at least forty nine different templates, we serve it by only fifteen templates (shown in Table 3.4). According to the results shown in Table 3.4 and 3.5), we conclude that, our proposed framework performs the same level of performance as an exhaustive system. It conforms completeness competency of this greedy template generation technique.

Furthermore, comparing information access performances over Linked Data with contemporary systems like GoRelations [40], SemSek [4], Alexandria [71], QAKiS [20] etc [67] and our proposed framework (shown in Table 3.8), we can see that our template generation technique works effectively. Therefore, we consider that our proposed framework can address the second research challenge comprehensively.

3.6 Summary

Because the use of keywords is a comfortable choice for data access, numerous researchers have worked to adapt keywords for use in Linked Data access. However, because keywords do not hold required Linked Data semantics, which are mandatory when performing Linked Data access, a number of researchers have worked on template-based access techniques that have the potential to bridge the gap between keywords and semantics. Until now, however, such initiatives have suffered from a lack of robust template establishment guidelines.

In this study, we proposed an outline for template construction, ranking, and merging that can be used for automatic keyword-based Linked Data access. This method utilizes the internal statistics of the data during the template construction ranking processes. We have also introduced a template merging technique that makes multi-depth query construction possible. Because our method relies on the internal statistics of data, which are calculated automatically, we believe BoTLRet provides a very promising tool for use in achieving fully automatic information access of Linked Data. We have also introduced a binary progressive processing paradigm that is scalable for any number of query keywords.

Other than defining resource-representing tags, i.e., *rtag* (see Section 3.1), no special customization is required to adapt our proposal to a Linked Dataset. Furthermore, the systems *rtag* include labels, titles, etc., which are quite generic among datasets. Experiments conducted using our proposed system have shown generally positive outcomes, which indicates that the system provides a functional technique for use in Linked Data access.

We presume that our template creation technique could benefit from use in conjunction with other Linked Data access approaches, such as automatic ontology inclusion, feedback incorporation, and other sophisticated keyword matching techniques that can provide more appropriate templates, and hope to explore these possibilities in our future work. As our work depends on various statistical parameters, we also presume that the incorporation of off-line processing could increase the system's performance. This, we feel, is another promising area for our future investigations.

Chapter 4

TLDRet: A Temporal Extension of BoTLRet

We first describe about its introduction (Section 4.1). Then we investigate how the temporal information usually are presented in the query (Section 4.2). It outlines how to identify temporal features in the input query. Then we describe the proposed temporal linked data retrieval framework TLDRet in details (Section 4.3). We show results of implementing our proposal through experimental results and discussion (Section 4.4). Later, we explain the proposed framework that how it solved the problem that existed in the contemporary systems (Section 4.5) Finally, in Section 4.6 we summarize the Chapter.

4.1 Introduction

Temporal features, such as an explicit date and time or a time-specific event, employ concise semantics for any kind of information retrieval. Therefore, temporal features should be suitable for linked data information retrieval. However, we have found that most linked data information retrieval techniques pay little attention to the power of temporal feature inclusion. We extend the basic information access framework BoTLRet so that it can incorporate temporal features and retrieve more concise Linked Data information. The evaluation of our system performance indicates that it is promising.

4.1.1 Motivation

Temporal feature related information, such as an explicit date and time or a time-specific event, is helpful for finding an appropriate result or for discovering a new relation. Though extraction of temporal feature related information of typical document-based data has quite a long history of research, the same kind of research for Linked Data has been comparatively less pursued. One reason could be that Linked Data have a different structure than typical document-based data. In this Chapter, we examine these issues and propose an efficient and easy-to-use Linked Data information access framework that can retrieve information related to temporal features.

Rula et al. investigated the 2011 Billion Triple Challenge (BTC) 2011 dataset and showed that some, but not many, contemporary Linked Data can store temporal fact related information [92]. The investigation by Rula et al. showed that contemporary Linked Data can store temporal fact related information according to two perspectives: i) document-centric, where time points are associated with the RDF triple, and ii) fact-centric, where time points or intervals are associated with facts [92]. Usually, document-centric temporal information is used to validate RDF triples, such as last date/time of modification of the triple. On the other hand, fact-centric temporal information provides a historical temporal value attachment to RDF resources, such as an example dataset that could store fact-centric temporal Linked Data for the *birthday of Michael Jackson* by a RDF triple <*res:Michael_Jackson prop:birthDate 29-Aug-1958*>. In this Chapter, we investigate temporal Linked Data information access from the fact-centric perspective. We also assume that inclusion of the document-centric perspective will be an interesting topic for us to explore in the future.

In Linked Data information access, temporal facts can be mentioned in various ways, such as the following:

- Data publishers could store temporal facts with various storage models, e.g., Temporal RDF [39] and stRDF [9], along with various time-related ontologies, e.g., Timely YAGO [110], owl-time[1], and timeline[2] [92].
- Data consumers could seek temporal facts with various structured time-specific queries, e.g., stSPARQL [9], or familiar keyword-based time-specific queries, such as explicit time-specific queries (e.g., *on the 29th of August 1958*) and event-specific queries (e.g., *during World War I*).

Because access of temporal Linked Data information requires handling all of those issues, it is a challenge for both data publishers and data consumers. The above issues motivated us to investigate the temporal feature related Linked Data information access.

4.1.2 Contributions

We extend BoLRet that hides complexities of temporal features of Linked Data and their handling. We contribute mainly by

(1.) proposing how to identify temporal features in the input query (describe in Section 4.2)

(2.) devising a Linked Data information access framework called **T**emporal **L**inked **D**ata **Ret**rieval extended framework (**TLDRet**) [85] that can explore temporal semantics (describe in Section 4.3)

(3.) developing and evaluating the system (described in Section 4.4)

[1]http://www.w3.org/TR/owl-time/
[2]http://motools.sourceforge.net/timeline/timeline.html

Table 4.1: Example questions and their corresponding comma-separated keywords

#	**Example Question (Q)**	**Corresponding Keywords**
1	Which music artist was born on the 29th of August 1958?	{music artist, birthday, on the 29th of August 1958}
2	Which US President was born during World War I?	{US President, birthday, during World War I}
3	Which US President started working 2 decades after the Iranian revolution started?	{US President, active years start date, after 2 decades of Iranian revolution, start date }
4	Which US President was born in the last century?	{US President, birthday, in last century}

To our knowledge, there are not many frameworks that can retrieve temporal feature related Linked Data information. From the Linked Data perspective, effective exploration of temporal semantics is an important advancement towards its realistic utilization.

4.2 Time & Event in Query

In this section, we investigate how the temporal information usually are presented in the query. It helps us to devise our retrieval framework, which is described in the next section.

Over the query, we will see some features that can turn query keywords to *temporal keywords* – keywords with time or events. Usually, the temporal feature attached sentences or queries always hold indicator words called *signal words* ([32, 36, 69, 93, 94]), as shown in Table 4.1. The table shows some example questions and their corresponding keywords. The first column shows question number, the second column shows the questions, and the third column question-corresponding keywords. We find that the Example Question #1 (Q#1) holds a temporal feature indicating the word "on" that follows the temporal feature "the 29th of August 1958", so the word "on" is considered a "signal word". Likewise, for the following example questions, the words "during", "after", "of", "in" are considered as *signal words*. Therefore, if we refer to the keywords holding temporal features as *temporal keywords*, we can identify them by the *signal word*. Usually, a *signal word* following the input keywords is the *temporal keyword*. The temporal keywords can be various in their types. Depending upon the types, they need to identify differently. Below we describe types of temporal keywords, and how they can be identified in details.

4.2.1 Types of temporal keywords

Over the text or sentence, the *temporal keyword* could hold a definite DATE/TIME (e.g, in Q#1). It also could represent some event information related words that could be converted to a definite DATE/TIME (e.g., in Q#2). Or, it could hold relative temporal values that will require some kind of adjustment before finding a definite DATE/TIME (e.g., in Q#3 and Q#4). The three types of *temporal keywords* are considered as the below:

(I.) *explicit temporal keyword* (Type-1 keyword)

(II.) *event temporal keyword* (Type-2 keyword)

(III.) *relative temporal keyword* (Type-3 keyword)

In the type categorization, the Type-1 and the Type-2 keywords both hold a single *signal word.* Thus, each of both types of *temporal keywords* represents a single temporal value. For example, the Type-1 keyword should belong to Q#1 i.e., "on the 29th of August 1958", and the Type-2 keyword should belong to Q#2 i.e., "during World War I".

On the other hand, the Type-3 keyword either holds two *signal words*, or a *signal word* and a temporal adverb. Thus, the Type-3 keyword contains two temporal values. In this case, we consider one temporal value as the *reference temporal value* and the other one as the *adjustment temporal value.* Here, the adjustment between the *reference temporal value* and the *adjustment temporal value* produces a single temporal value. For example, the Type-3 keyword should belong to Q#3 and Q#4, and they are "after 2 decades of Iranian revolution start date" and "in last century", respectively. In the Q#3, the *reference temporal value* is "Iranian revolution start date" and the *adjustment temporal value* is "after 2 decades".

However, in some cases, the *relative temporal keyword* i.e., Type-3 keyword might not explicitly hold the *reference temporal value.* Depending on whether the *relative temporal keyword* holds an explicit *reference temporal value*, we classify the *relative temporal keyword* into two sub-types:

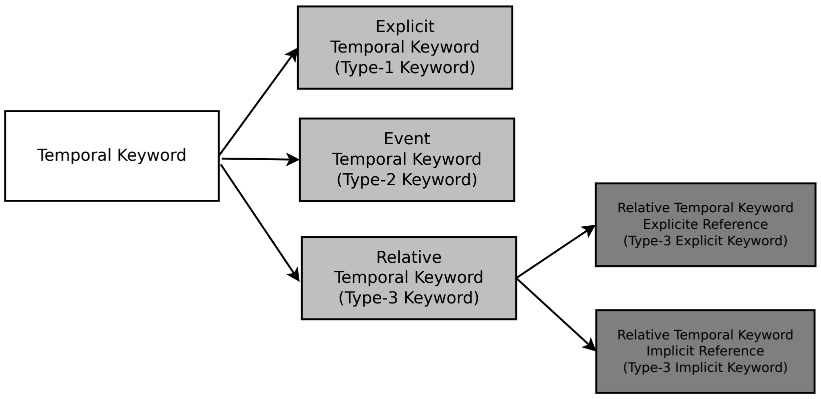

Temporal Keyword
Explicit Temporal Keyword (Type-1 Keyword)
Event Temporal Keyword (Type-2 Keyword)
Relative Temporal Keyword (Type-3 Keyword)
Relative Temporal Keyword Explicite Reference (Type-3 Explicit Keyword)
Relative Temporal Keyword Implicit Reference (Type-3 Implicit Keyword)

FIGURE 4.1: Types and sub-types of the *temporal keyword*

(i.) *relative temporal keyword - explicit reference* (Type-3 explicit keyword)

(ii.) *relative temporal keyword - implicit reference* (Type-3 implicit keyword)

The Type-3 explicit keyword holds two *signal words*. The second *signal word* following the input keywords hold the *reference temporal value*. On the other hand, the Type-3 implicit keyword holds a *signal word* and a temporal adverb. The temporal adverb implicitly holds both the *reference temporal value* and the *adjustment temporal value*. The Q#3 holds sub-type (i.) type *relative temporal keyword* i.e., Type-3 explicit keyword. In the Q#3, the word "after" is the first *signal word* and the word "of" is the second *signal word*. The second *signal word*, "of", following the input keywords, "Iranian revolution start date", is used to capture the *reference temporal value*, while the first *signal word*, "after", following the input keywords, "2 decades", is used to capture the *adjustment temporal value*. On the other hand, the Q#3 holds sub-type (ii.) type *relative temporal keyword* i.e., Type-3 implicit keyword. In the Q#4, the word "in" is the single *signal word* and the following input keywords, "last century", are the temporal adverb. This temporal adverb is used to capture both the *reference temporal value* and the *adjustment temporal value*. Figure 4.1 shows types and sub-types of the *temporal keyword*. Our type categorization comply Saquete et.al., type categorization ([93]). However, we categorize the Type-3 keyword more fine grained. It supports to capture more in-depth semantics.

4.2.2 Technical basis of identifying temporal keywords

Temporal keyword types presents their identification basis. To identify *temporal keywords* over the query, we consider the below two criteria:

- how many *signal word*(s) a *temporal keyword* hold(s)?
- how the *temporal keyword* can be converted to its “normalized output”?

If a *temporal keyword* holds only one *signal word*, it either can be a Type-1 keyword or a Type-2 keyword . In such a case, we identify the *temporal keyword* by the normalized output. A language parser is used over the input keywords to find the normalized output. If the normalized output of the *signal word* following the input keywords holds DATE/TIME, the *temporal keyword* belongs to the Type-1 keyword ; otherwise, it belongs to the Type-2 keyword .

On the other hand, if the *temporal keyword* holds two *signal words* or a *temporal keyword* and a temporal adverb, it belongs to the Type-3 keyword. If the normalized output of the (first) *signal word* following the input keywords holds NUMBER/DURATION and the next *signal word* following the input keywords resembles the Type-2 keyword , we identify this Type-3 keyword as the Type-3 explicit keyword; otherwise, we identify this Type-3 keyword as the Type-3 implicit keyword.

4.3 TLDRet: Linked Data Retrieval Framework with Temporal Semantics

TLDRet is our proposed system[3]. TLDRet takes keywords as input and generates possible information as output. It captures input keywords’ temporal semantics. The *signal words* are used to capture those temporal semantics. In the TLDRet, we compile *signal words* from ([32, 36, 93]). Moreover, to annotate temporal value of temporal keywords, we adapt TIMEX3 DATE/TIME annotation.

TLDRet uses BoTLRet as the basic Linked Data information retrieval framework. We have already described BoTLRet in Chapter 3. Figure 4.2 shows the overall process flow for the TLDRet. It has two phases:

(1.) phase 1 - query text processing. Here we analyze the input keywords and annotates the temporal value of *temporal keywords* to TIMEX3.

[3]an extension of the previous framework TLDRet_old ([85]).

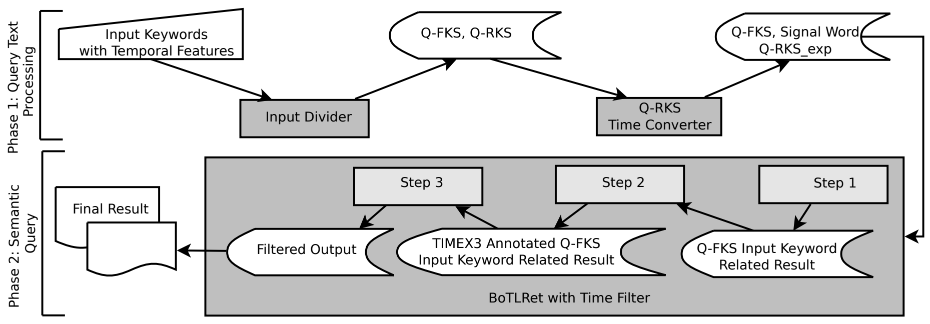

Phase 1: Query Text Processing
Input Keywords with Temporal Features
Q-FKS, Q-RKS
Q-FKS, Signal Word Q-RKS_exp
Input Divider
Q-RKS Time Converter
Phase 2: Semantic Query
Final Result
Step 3
Step 2
Step 1
Filtered Output
TIMEX3 Annotated Q-FKS Input Keyword Related Result
Q-FKS Input Keyword Related Result
BoTLRet with Time Filter

FIGURE 4.2: Overall work flow of our proposed method

(2.) phase 2 - semantic query. Here we impose a time filter to produce the intended result.

The below we describe these two phases in details.

4.3.1 Phase 1: Query text processing

Here we describe about phase 1, query text processing. This phase performs pre-processing tasks before the inclusion of temporal semantics. In this phase, we analyze the input keywords to find the *temporal keywords* and their types. Then we annotate the temporal value of *temporal keywords* by their corresponding TIMEX3 annotation. The *input divider* process and the *Q-RKS time converter* process accomplish these tasks. We describe the processes below.

4.3.1.1 The input divider process

The *input divider* process divides input keywords. This process finds the *temporal keywords* and *non-temporal keywords* and classifies the *temporal keywords* with their types.

From the perspective of answering the temporal question, Saquete et al. showed a technique using an *ordering key* ([93]). The *ordering key* preserves the temporal semantics of input keywords by introducing some kind of information validity constraint (details are described in the following phase). The *ordering key* needs to divide input keywords into two keyword sets. The two keyword sets are as follows.

1. *Q-Focus keyword set* (Q-FKS): input keywords that specify the information that the user is searching. The *signal word* prior input keywords fall into Q-FKS.
2. *Q-Restriction keyword set* (Q-RKS): input keywords that specify temporal features used to restrict required information. The *signal word* and its follower input keywords fall into Q-RKS. In TLDRet, *temporal keywords* belong to Q-RKS.

If we execute the *input divider* process over the example questions shown in Table 4.1, we get the following Q-FKSs and Q-RKSs:

- Q#1: Q-FKS = {"music artist, birthday"}, Q-RKS = {"on the 29th of August 1958"}
- Q#2: Q-FKS = {"US President, birthday"}, Q-RKS = {"during World War I"}
- Q#3: Q-FKS = {"US President, active years start date"}, Q-RKS = {"after 2 decades of Iranian revolution start date"}
- Q#4: Q-FKS = {"US President, birthday"}, Q-RKS = {"in last century"}

As mentioned, Q-RKS input keywords hold *temporal keywords* that can be classified into 4 different types – Type-1 keyword, Type-2 keyword, Type-3 explicit keyword, and Type-3 implicit keyword. By using the *temporal keyword* identifying technique (described in the previous section), the *input divider* process classifies the types of Q-RKSs.

4.3.1.2 The Q-RKS time converter process

Whatever the types, the *Q-RKS time converter* process finds the definite DATE/-TIME for Q-RKS in the TIMEX3 format. Below we describe the process for finding the definite DATE/TIME for each type of *temporal keyword.*

- ***Explicit temporal keyword***: If Q-RKS input keywords hold the Type-1 keyword , i.e., keywords with a date and time, the *Q-RKS time converter* generates its definite DATE/TIME in TIMEX3 format by parsing the *signal* *word* following part of the Type-1 keyword . We refer to this TIMEX3 value as Q-RKS_exp.

The Q#1 belongs to such a case, where the *Q-RKS time converter* process converts the *signal word* following part of the Type-1 keyword (i.e., "the 29$^{\text{th}}$ of August 1958") to their corresponding TIMEX3 value as 1958-08-29.

- ***Event temporal keyword***: We use BoTLRet to convert the *event temporal keyword* to its definite DATE/TIME. So, if Q-RKS input keywords hold the Type-2 keyword , we first execute BoTLRet over the *signal word* following part of the *event temporal keyword.* Then, by using a language parser, the output of BoTLRet is parsed. Over the parsed output, we capture the DATE/TIME/DURATION type normalized NER (Named Entity Recognizer) value. These NER values are considered as definite DATE/-TIME of *event temporal keyword.* We convert the NER values in TIMEX3 format and refer as *event temporal keyword* corresponding Q-RKS_exp.

  For example, the Q#2 belongs to a case where the *Q-RKS time converter* converts the Type-2 keyword to the corresponding TIMEX3 values. The values are 1914-07-28, 1919-06-28, 1919-09-10, 1919-11-27, 1920-06-04, and 1920-08-10. Each of these dates are somehow related to "World War I": date 1914-07-28 is related to the armistice with Germany, date 1919-06-28 is related to signing the treaty of Versailles, and so on.

- ***Relative temporal keyword***: Since the Type-3 keyword holds two temporal values at the same time, finding the definite DATE/TIME requires some kind of adjustment. Therefore, to find the definite DATE/TIME for a Type-3 keyword, we retrieve its *reference temporal value* and *adjustment temporal value.* We adjust the *reference temporal value* by adding or subtracting the *adjustment temporal value*, which gives us the definite DATE/TIME. Below, first we describe the retrieval of the *reference temporal value* and the *adjustment temporal value.* Then, we describe their adjustment.

  – **Retrieval of the *reference temporal value* and the *adjustment temporal value***: To find the *reference temporal value* from the Type-3 explicit keyword, we execute the *Q-RKS time converter* process over the second *signal word* following part of the keywords. This generates a TIMEX3 value that is used as the *reference temporal value.* On the other hand, to find the *adjustment temporal value* from the

Type-3 explicit keyword, we execute a language parser over the keywords between the first *signal word* and the second *signal word*. From the parsed output, we collect the NER values for the DURATION/-DATE/TIME/NUMBER type NER, and then convert the NER values as TIMEX3. However, we find that some language parsers, though able to identify the DURATION type NER, cannot generate a TIMEX3 value for the DURATION type NER. In such a case, we also execute the Tarsqi Toolkit[4] over the keywords between the first *signal word* and the second *signal word*. The Tarsqi Toolkit confirms any missing DURATION type NER.

For example, for the Type-3 explicit keyword of the Q#3, the temporal keyword is "after 2 decades of Iranian revolution start date", the first *signal word* is "after" and the second *signal word* is "of". The execution of the *Q-RKS time converter* process over the keywords "Iranian revolution start date" generates the TIMEX3 *reference temporal value* as 1978-01-07. On the other hand, the execution of the Stanford language parser ([22]) over the keywords "2 decades" can identify NER as the DURATION type, which should be the *adjustment temporal value*. But the parser cannot generate the TIMEX3 value. Therefore, execution of the Tarsqi Toolkit over the keywords "2 decades" generates the TIMEX3 value "P20Y". By the TIMEX3 convention, this means 20 years duration, where "P" indicates duration and "Y" indicates year.

For the Type-3 implicit keyword, both the *reference temporal value* and the *adjustment temporal value* come from its temporal adverb. In such a case, the *reference temporal value* is always the current date and time. To find the *reference temporal value*, we convert the current date and time into their TIMEX3 value. To find the *adjustment temporal value*, we execute a language parser over the *signal word* following the keywords. Here, finding the *adjustment temporal value* in TIMEX3 is the same as described in the above paragraph. For example, for the Type-3 implicit keyword of the Q#4, we get the *reference temporal value* as 2016-03-05[5] and the *adjustment temporal value* as "P100Y".

– **Getting the definite DATE/TIME**: After finding the *reference temporal value* and the *adjustment temporal value*, we adjust them to get

[4]http://www.timeml.org/site/tarsqi/toolkit/

[5]Date of execution 5th March 2016.

the definite DATE/TIME, Q-RKS_exp. The adjustment between the *reference temporal value* and the *adjustment temporal value* is carried out by the (first) *signal word* of the Type-3 keyword and the guideline of the TIMEX3 time convention[6]. If the first *signal word* means "before", we subtract the *reference temporal value* and the *adjustment temporal value*; if it means "after" we add these values. For example, as the definite DATE/TIME of the Q#3 Q-RKS ("after 2 decades of Iranian revolution start date"), we get 1998-01-07. Here the first *signal word* after the *reference temporal value* is 1978-01-07 and the *adjustment temporal value* is "P20Y". Therefore, adding 1978-01-07 and "P20Y" generates the TIMEX3 value as 1998-01-07.

Finally, as the end result of phase 1, we get keywords for the Q-FKS and the definite DATE/TIME for the Q-RKS (Q-RKS_exp).

Next, we talk about phase 2 of the TLDRet, which generates the final result. If it is found that the input keywords only belong to Q-RKS in phase 1, the output of the *Q-RKS time converter process* is considered as the final result. In such a case, we do not require phase 2.

4.3.2 Phase 2: Semantic query

This subsection describes phase 2, the semantic query. In this phase, we attach Q-RKS_exp to the output of Q-FKS in such way that the temporal semantics of the input keywords are preserved. The *ordering key* framework ([93]) supports the preservation of temporal semantics. By preserving the temporal semantics, we filter out the output of Q-FKS input keywords that retrieve a final result. The *BoTLRet with time filter* process accomplishes the above tasks.

However, before describing the *BoTLRet with time filter* process, we should introduce the basics of the *ordering key*. The *ordering key* defines the constraint of information validity constructed for three parameters: i) *signal word*, ii) temporal feature related part of Q-FKS, and iii) temporal feature related part of Q-RKS. With this constraint, the *ordering key* incorporates the temporal semantics of input keywords that eventually will give the option of information filtering. For

[6]http://www.timeml.org/tempeval2/tempeval2-trial/guidelines/timex3guidelines-072009.pdf

TABLE 4.2: Example of *signal words* and *ordering keys*

Signal word	**Ordering key**
Before	F1 < F2
On, In, When, At the same time	F1 = F2
After, Since	F1 > F2
During, While, For	F2i <= F1 <= F2f
From, About	F2 <= F1 <= F3

every *signal word*, the *ordering key* introduces the constraint of information validity. For the Q#1 where the *signal word* is "on", the temporal feature related part of Q-FKS is "birthday" and the temporal feature related part of Q-RKS is "the 29th of August 1958", so the constraint of information validity is "birthday" = "the 29th of August 1958".

For simplicity, we represent the temporal feature related part of Q-FKS as symbol F1 and the temporal feature related part of Q-RKS as symbol F2. Table 4.2 shows some *signals words* and their *ordering keys*. In this table, the first column shows the *signal word* and the second column shows the corresponding *ordering key* between F1 and F2. To represent the time interval, F2 is stated as its initial (F2i) and final (F2f) points. Some of them explicitly attach the temporal feature related part of Q-RKS to two points, F2 and F3. The last two rows of the table show the time interval.

Now we describe the *BoTLRet with time filter* process. This process takes Q-FKS, the *signal word*, and Q-RKS_exp as input and generates the final output. This process has three steps.

(I.) In Step 1, we use our BoTLRet over the Q-FKS input keywords to find the Q-FKS related result

(II.) In Step 2, we parse the Step 1 result and replace the NER values of the DATE/TIME/DURATION type by their corresponding TIMEX3 values

(III.) In Step 3, according to the *signal word* corresponding to the *ordering key*

we filter the TIMEX3 annotated result of Step 2 that complies with the temporal semantics. This filtered result is considered as our final output.

4.3.2.1 Step 1

As mentioned, Q-FKS input keywords specify the information that the user is searching for. Step 1 generates such information. We employ BoTLRet over the Q-FKS, which generates all such possible information irrespective of the temporal feature being related or non-related.

4.3.2.2 Step 2

From the temporal linked data information retrieval perspective, Q-FKS always holds the temporal feature related part used in information filtering. As mentioned, we present the temporal feature related part by symbol F1. Step 2 finds this temporal feature related part and annotates its temporal value by its TIMEX3 annotation so that the annotated value can be used to maintain the temporal semantics. To do this, we take the output of Step 1 and parse it, then annotate the NER values of the DATE/TIME/DURATION type by their corresponding TIMEX3 values.

4.3.2.3 Step 3

Step 3 is the last step, which retains the temporal semantics of the input keywords and retrieves the final result. According to the constraint of information validity (*ordering key*), the temporal feature related part of Q-FKS is represented by F1, and the temporal feature related part of Q-RKS is represented by F2. We filter the TIMEX3 annotated result of Step 2, which gives the final result. Below we describe each step in detail.

As mentioned, to comply with the constraint of information validity, we assign

- the temporal feature related part of Q-FKS to symbol F1
- the temporal feature related part of Q-RKS to symbol F2

Therefore, in information filtering, we require the TIMEX3 value of F1 and the TIMEX3 value of F2 to implement the temporal semantics.

The TIMEX3 value of F1 comes from the TIMEX3 annotated result of Step 2. Here, the NER value for the DATE/TIME/DURATION type is the TIMEX3 value of F1. For example, if the TIMEX3 annotated result of Step 2 is "Beethoven ... 1770-12-16", then the TIMEX3 value of F1 is 1770-12-16. Whereas, the TIMEX3 value of F2 comes from Q-RKS_exps.

In our observation, in some cases, we can get multiple Q-RKS_exps. Usually, the multiple Q-RKS_exps could be generated to capture an intervals such as one for Type-1 keyword , "from 28th July 1914 to 10th August 1920", or *event temporal keywords*, "during World War I". Or, they could be generated when Q-RKS input keywords are not enough to define a single Q-RKS_exp. In such a case, according to the *ordering key*, we decide the *temporal picking point* (TPP) for the temporal feature related part of Q-RKS. This TPP helps in picking the best possible Q-RKS_exp among many. For the interval, the TPP is considered for two parts: initial TPP (TPPi) and final TPP (TPPf).

Table 4.3 shows the TPP picking strategy for various *ordering keys*. In the first column, the *ordering key type* divides *ordering keys* into two types: *point of time* and *interval*. The second and third columns show *ordering keys* and their corresponding *temporal picking points*, respectively, which describe how we pick the Q-RKS_exp value. The *point of time* type *ordering keys* select a single Q-RKS_exp. The *interval* type *ordering keys* select two Q-RKS_exps: initial and final. That is, over an example dataset, for the Q#1, the TPP is 1958-08-29. This is because the *ordering key* of the Q#1 is F1 = F2, which finds only one TIMEX3 annotation for the temporal feature related part of Q-RKS. On the other hand, over the same example dataset, for the Q#2, the TPPi is 1914-07-28 and the TPPf is 1920-08-10. This is because the *ordering key* of the Q#2 is F2i<=F1<=F2f , where we get multiple Q-RKS_exps, e.g., 1914-07-28, 1919-06-28, 1919-09-10, 1919-11-27, 1920-06-04, and 1920-08-10, among which the lowest value is 1914-07-28 and the highest value is 1920-08-10.

To filter the TIMEX3 annotated result of Step 2, we find the TIMEX3 value of F1, the TPP value, and construct the constraint of information validity according to the *ordering key*. Then we pick the result of Step 2 that passes the constraint of information validity. That is how we preserve temporal semantics and generate a concise result.

TABLE 4.3: *Ordering keys* and their *temporal picking points*

Ordering key type	**Ordering key**	**Temporal Picking Point (TPP/TPPi/TPPf)**
Point of time	F1 < F2	Pick lowest Q-RKS_exp value among all such values as TPP
	F1 = F2	Pick every Q-RKS_exp value as TPP
	F1 > F2	Pick highest Q-RKS_exp value among all such values as TPP
Interval	F2i <= F1 <= F2f	Pick lowest Q-RKS_exp value among all such values as TPPi
		Pick highest Q-RKS_exp value among all such values as TPPf
	F2 <= F1 <= F3	Pick Q-RKS_exp value for TIMEX3 converted value of F2 as TPPi
		Pick Q-RKS_exp value for TIMEX3 converted value of F3 as TPPf

Table 4.4 shows the different execution steps of the *BoTLRet with time filter* process for the Q#1 over the example dataset. Here, the first column shows the step number, the second column shows the step description, and the third column shows the step result. The execution of Step 1 selects all possible results for Q-FKS input keywords, {*music artist, birthday*}. In Step 2, the execution of the parser over the Step 1 result annotates the temporal value attached parts by the corresponding TIMEX3 values. Q#1 holds *on* as the *signal word*. So, in Step 3, by using the *signal word* corresponding to the *ordering key*, the *ordering key* is F1 = F2, which finds the TPP value as 1958-08-29. Finally, by considering the constraint of information validity, the *BoTLRet with time filter* process filters out the possible result of the Q#1.

In the previous illustration, the temporal feature related part of the Q-FKS input keywords (represented by F1) corresponds to a single point of time ("birthday"). However, in some cases, the temporal feature related part could correspond to multiple points of times. This is because Q-FKS input keywords could hold multiple dates and time associated parts, or an associated part could hold an interval. That is, for example, for input keywords {"US President", "serving period", "during World War I"}, the Q-FKS input keywords are {"US President", "serving period"}, where the date and time associated part is "serving period", which is an interval. In such a case, we consider all possible TIMEX3 values of F1, construct all possible constraints of information validity and filter the possible information.

Table 4.4: Semantic query scenario for Q#1 (Example Question #1)

Step	**Step description**	**Result**
1	Execution of BoTLRet over Q-FKS input keywords {*music artist, birthday*}	Beethoven ... December, 16, 1770
		Iikka Paananen ... 29th December 1960
		Michael Jackson ... August 29, 1958
		Ramona Maria Luengen ... December, 29, 1960
		... ...
2	Execution of parser over Step 1result for DATE/ TIME/DURATION type NER values	Beethoven ... 1770-12-16
		Iikka Paananen ... 1960-12-29
		Michael Jackson ... 1958-08-29
		Ramona Maria Luengen ... 1960-12-29
		... ...
3	Use of TIMEX3 value of F1, the TPP value, and constraint of information validity	Michael Jackson ... 1958-08-29

4.4 Experiment

Like the basic framework BoTLRet, we also use the QALD[7] (Question Answering over Linked Data) open challenge question sets in our experiment. As we know, the QALD open challenge includes natural language question sets from DBPedia and MusicBrainz datasets. Until recently, it contains five different challenges − QALD-1, QALD-2, QALD-3, QALD-4, and QALD-5. Each of the challenge question sets are further divided into a training question set and a test question set. Among the five challenge sets, first two − QALD-1 and QALD-2 − focus on English natural language questions. On the other hand, last three sets focus on multilingual questions. Moreover, we found that QALD-1 and QALD-2 hold more number of temporal questions than the latter sets. Furthermore, we found that the few temporal questions that belong to QALD-3, QALD-4, and QALD-5 are similar to the questions of QALD-1 and QALD-2. Therefore, we experimented TLDRet for the QALD-1 and QALD-2 questions.

Since our proposed system TLDRet depends on keywords, we intuitively retrieve keywords that are simultaneously present in the datasets and mean the query questions. We evaluate the TLDRet for the efficiency achieved in the temporal feature related question (or temporal question) answering.

[7]http://greententacle.techfak.uni-bielefeld.de/˜cunger/qald/

For temporal question answering, we evaluate the TLDRet for QALD questions that relate to temporal features in their answers[8]. An example of such a question is "Which artists were born on the 29th of December 1960?" (MusicBrainz QALD-2 test Q.# 13).

In the evaluation, first we evaluate the TLDRet for its performance. Second, we evaluate its efficiency with other systems.

In the TLDRet, output normalization and TIMEX3 annotation are done by the Stanford parser ([22]).

4.4.1 Performance over the QALD temporal questions

In the performance measurement, we check whether the proposed system is able to retrieve information over QALD-1 and QALD-2 temporal questions. We evaluate the performance for question sets from each participant dataset (DBpedia and MusicBraiz), and then delve into this evaluation for each type of *temporal keyword* question.

4.4.1.1 Question set performance

In this performance measurement, we check how effective the TLDRet is in temporal question answering.

We execute all temporal feature related QALD-1 and QALD-2 questions for both DBpedia and MusicBrainz datasets. We evaluate each set of questions and check average precision, average recall, and average F1 measure for each set. For each dataset, although QALD-1 and QALD-2 questions are divided into training questions and test questions, which we do not consider them as different question sets, because it increases the number of questions in each set. To calculate precision, recall, and F1 measure, the QALD organizers give gold standard results for each question. We use them to calculate the performance. For each participant question set, we calculate the average performance by summing up the performance of

[8] The questions are **DBpedia QALD-1 training Q.#** 8 and **test Q.#** 11, 24, 41; **DBpedia QALD-2 training Q.#** 2, 37, 39, 52, 90, 91 and **test Q.#**7, 12, 25, 56, 71, 74, 92, 94; **MusicBrainz QALD-1 training Q.#** 3, 6, 10, 12, 13, 14, 21, 23, 26, 28, 30, 31, 41, 49, 66, 72, 77, 89, 100 and **test Q.#** 1, 3, 7; **MusicBrainz QALD-2 training Q.#** 3, 10, 12, 13, 14, 21, 23, 26, 28, 31, 67, 72, 77, 89, 99, 100 and **test Q.#** 1, 3, 7, 16.

Participant question set	**# of questions**	**Performance of TLDRet**		
		Precision	Recall	F1 Measure
DBPedia QALD-1	4	1.00	1.00	1.00
QALD-2	14	0.93	0.93	0.93
	Average	0.95	0.95	0.95
MusicBrainz QALD-1	22	0.68	0.68	0.68
QALD-2	20	0.75	0.75	0.75
	Average	0.71	0.71	0.71

Table 4.5: QALD-1 and QALD-2 temporal question answering performance by TLDRet

the corresponding questions of the question set and dividing them by the number of questions held by each participant question set.

Table 4.5 shows the performance of TLDRet for QALD-1 and QALD-2 temporal question answering. Here, the first column shows the source of the participant question set, the second column shows the number of questions in each participant question set, and the third column shows the performance of TLDRet. Moreover, for each dataset, we calculate the average performance.

We find that the TLDRet achieves high performance over the DBPedia dataset questions. In contrast, the TLDRet achieves lower performance over the MusicBrainz dataset questions. Our detailed investigation finds that this performance drop is not because of a lack of the temporal feature attachment. Rather, BoTLRet is not able to generate Q-FKS keyword related information, which is a prerequisite for temporal feature attachment. The BoTLRet depends on some fixed structure templates which, in some cases, fail to resemble the dataset. To solve this problem, BoTLRet needs to consider templates other than the fixed structures. If this problem is solved, the TLDRet will also be able to achieve better performance on MusicBrainz dataset questions.

We adapt temporal semantics into the keyword-based QA system. According to the temporal feature related QALD-1 and QALD-2 questions, we conclude that the TLDRet can retrieve temporal feature related information over the linked data comprehensively. Moreover, unlike the system described in ([109]), the TLDRet works in any domain. We should also acknowledge that the TLDRet is developed on top of BoTLRet [86], which uses exactly matching keywords as input. These exactly matching keywords facilitate the TLDRet in achieving the same recall and precision values in the experiments, because the TLDRet never gets any false-positive results.

In the TLDRet, successful retrieval of temporal feature related information over linked data occurs due to symbiotic adaptation of the *temporal keyword* and the *ordering key* for the given input keywords. We explore all temporal values, whether they are mentioned explicitly (e.g., by explicit date and time) or implicitly (e.g., by event-specific input), in a common format that helps to filter out concise information. Though the linked data hold a different structure than the document-based data, due to the TLDRet we propose a technique that can capture temporal semantics over the linked data. This would be similar to the temporal semantics adaptation technique in standard document-based data.

4.4.1.2 Performance of each type of *temporal keyword* question answering

In this performance measurement, we check how effective the TLDRet is over each type of *temporal keyword* question answering.

To do this, we classify the QALD-1 and QALD-2 temporal questions with their corresponding *temporal keyword* types. Then, for each question set, we evaluate the average precision, average recall, and average F1 measure for the same type of questions.

Table 4.6 shows type-wise question answering performance for the QALD-1 and QALD-2 temporal questions. Here, the first column shows the source of the participant question set, the second column shows the type of *temporal keyword* question, the third column shows the number of questions in each participant question set, and the fourth column shows the performance of TLDRet.

We find that the TLDRet can retrieve information for all three types of *temporal keyword* corresponding questions. In the TLDRet, successful retrieval of information for each type of *temporal keyword* question depends on effectively identifying the different temporal keyword types and finding the type-wise definite DATE/-TIME.

Table 4.6: Type-wise question answering performance for QALD-1 and QALD-2 temporal questions by the TLDRet.

Participant question set	**Type**	**# of Qs**	**Performance of TLDRet**		
			Precision	Recall	F1 Measure
DBPedia QALD-1	Type-1	2	1.00	1.00	1.00
	Type-2	2	1.00	1.00	1.00
	Type-3	0	-	-	-
DBPedia QALD-2	Type-1	3	1.00	1.00	1.00
	Type-2	9	0.89	0.89	0.89
	Type-3	2	1.00	1.00	1.00
MusicBrainz QALD-1	Type-1	5	0.60	0.60	0.60
	Type-2	17	0.69	0.69	0.69
	Type-3	0	-	-	-
MusicBrainz QALD-2	Type-1	6	0.83	0.83	0.83
	Type-2	14	0.71	0.71	0.71
	Type-3	0	-	-	-

4.4.2 Efficiency comparison with other systems

In the efficiency comparison with other systems, we compare whether the proposed system performs better in retrieving temporal feature related information. We compare the TLDRet with QALD-2 open challenge participant systems [67] and the TLDRet_old [85]. The TLDRet is our previous framework, which can retrieve temporal questions. However, it cannot retrieve information for the Type-3 keyword type questions.

We also delve into the efficiency comparison by the types of *temporal keyword* questions. It shows how the systems perform over the different *temporal keyword* type questions.

4.4.2.1 Comparison with other systems

In this efficiency comparison, we check how much better TLDRet is over the other linked data information access systems. We compare the efficiency only for temporal question answering.

For temporal questions, we compare the TLDRet with the QALD-2 open challenge participant systems named SemSek ([4]), Alexandria ([71]), MHE, and QAKiS

Table 4.7: Performance comparison among the TLDRet, the QALD-2 open challenge participant systems, and the TLDRet_old for the temporal feature related DBpedia QALD-2 test questions

System	**Average Precision**	**Average Recall**	**Average F1 Measure**
SemSeK	0.25	0.25	0.25
Alexandria	0.00	0.00	0.00
MHE	0.25	0.25	0.25
QAKiS	0.00	0.00	0.00
TLDRet_old	0.63	0.63	0.63
TLDRet	**0.88**	**0.88**	**0.88**

([20]), and the TLDRet over DBPedia test questions[9]. Because of the availability of the challenge participant systems' results for the DBPedia QALD-2 test question set, we use this question set for comparison. We find the DBPedia QALD-2 test set holds eight temporal questions (Q. # 7, 12, 25, 56, 71, 74, 92, 94), so we collect each question's precision, recall, and F1 measure for the challenge participant systems (SemSek, Alexandria, MHE, and QAKiS). We consider the precision, recall, and F1 measure as 0 (zero) if any system does not participate in question answering. We execute the TLDRet_old and the TLDRet for the DBPedia QALD-2 test question set. For each system, we calculate average precision, average recall, and average F1 measure.

Table 4.7 shows the performance comparison (average precision, average recall, and average F1 measure) among the TLDRet, the QALD-2 open challenge participant systems [67], and the TLDRet_old [85]. It is seen that our TLDRet decisively outperforms the challenge participant systems.

Again, we conclude that our proposed method successfully adapts the *temporal keyword* and the *ordering key*, and explores all temporal values for a common annotation that filters out possible information efficiently. Since our proposed system TLDRet adapts a temporal semantics adaptation technique similar to our standard document-based data and yet ignored by the challenge participant systems, our systems outperforms the challenge participant systems.

[9]http://greententacle.techfak.uni-bielefeld.de/~cunger/qald/2/dbpedia-test-questions.xml

TABLE 4.8: Performance of each *temporal keyword* type of question for the SemSeK, the MHE, the TLDRet_old, and the TLDRet systems.

System	**Type**	**# of Qs**	**Performance of TLDRet**		
			Precision	Recall	F1 Measure
SemSeK	Type-1	1	0.00	0.00	0.00
	Type-2	5	0.40	0.40	0.40
	Type-3	2	0.00	0.00	0.00
MHE	Type-1	1	0.00	0.00	0.00
	Type-2	5	0.40	0.40	0.40
	Type-3	2	0.00	0.00	0.00
TLDRet_old	Type-1	1	1.00	1.00	1.00
	Type-2	5	0.80	0.80	0.80
	Type-3	2	0.00	0.00	0.00
TLDRet	Type-1	1	1.00	1.00	1.00
	Type-2	5	0.80	0.80	0.80
	Type-3	2	1.00	1.00	1.00

4.4.2.2 Comparison for different types of *temporal keyword* questions

In this efficiency comparison, we compare four linked data information access systems to see how they perform over different *temporal keyword* type questions. It gives us insight about each system's capability.

We classify questions from the QALD-2 DBpedia test set into their corresponding *temporal keyword* types. Then, for the each system, we evaluate average precision, average recall, and average F1 measure of the same type of questions.

Table 4.8 shows the performance of each *temporal keyword* type of question for the SemSeK, the MHE, the TLDRet_old, and the TLDRet systems. Here, the first column shows the name of the system, the second column shows the type of *temporal keyword* question, the third column shows the number of questions in each type of *temporal keyword* question, and the fourth column shows each system's average performance for each type of question. We ignore the performance of the Alexandria and the QAKiS because they cannot answer any of the questions.

We find that the TLDRet is the only system that can retrieve information for all three types of *temporal keyword* questions and so the experiment validates its effectiveness. Particularly, for the Type-3 keyword type questions, getting the gold standard results validates the procedure for finding the definite DATE/TIME. This procedure adjusts the *reference temporal value* and the *adjustment temporal value*. The two QALD-2 open challenge participant systems only can answer the

Type-2 keyword type questions. On the other hand, the TLDRet cannot answer the Type-3 keyword type questions.

In the TLDRet, successful retrieval of information for each type of *temporal keyword* questions depends on effectively identifying the different temporal keyword types and finding the definite DATE/TIME of each type.

4.5 Discussion

In Linked Data information access, TLDRet addresses the last research challenge that we described in Chapter 1 Section 1.2.1. That is about

- "Temporal Lacking" where we described that although information access can be facilitated by temporal features, such as a date and time or a time-specific event, contemporary Linked Data information access systems fail to capture them. As a result, accessing of Linked Data should capture the temporal semantics.

The below we discuss about how TLDRet addresses the challenge.

- the proposed framework TLDRet incorporates temporal features of query, which addresses the last challenge stated the above. To our knowledge, TLDRet is one of a first systems over Linked Data that retrieves Linked Data information for various type of temporal features. It can handle explicit temporal feature such as definite DATE/TIME, event followed temporal feature such as "World War I" and relative temporal feature such as "last century". Therefore, TLDRet performs better in temporal feature related question answering tasks. Table 4.7 in Chapter 4 shows this comparison result where we see that proposed framework out-performs other contemporary systems SemSek [4], Alexandria [71], MHE, QAKiS [20] etc [67]. Therefore, we consider that our proposed framework TLDRet can address the third research challenge that we stated in Chapter 1 Section 1.2.1, effectively.

4.6 Summary

Linked data currently hold a very large amount of knowledge. Since linked data knowledge is connected knowledge, an efficient retrieval framework can generate more semantically enriched information. Usually, temporal values such as date and time or time of an event enrich semantics and help in finding new information or new links. However, current linked data information retrieval studies pay less attention to temporal semantics for linked data information retrieval. We focus on this issue and propose a linked data retrieval framework with temporal semantics.

Like the temporal semantics adaptation technique in usual document-based data, our proposed system TLDRet adapts a *temporal keyword*, an *ordering key* over linked data and captures the required temporal semantics. We outline that how temporal features can be identified in the query which helped to adapt the temporal semantics. With the outline, we explore all temporal values, whether they are mentioned explicitly or implicitly in the query, for a common annotation that helps to filter out concise information. The the basic Linked Data information retrieval system BoTLRet retrieve information for keywords that are not related with temporal semantics. Later, we filter out some information that abide by the query's temporal semantics. Our proposed method is not confined to a particular domain.

An experiment shows the efficiency of our proposed system. Our study is influenced by Saquete et al. in ([93]). Saquete et al. advocated adapting any general-purpose QA system to find the temporal feature related part of a question. However, finding the temporal feature related part of question is not easy and is not always consistent. Since linked data can hold structured data that are different than standard document-based data, linked data information retrieval requires different treatment. Therefore, we concluded that we propose a linked data information retrieval framework that can capture the temporal feature related part of question consistently. From the linked data perspective, retaining the temporal semantics (both for *explicit temporal keywords* and *implicit temporal keywords*) is an important advancement towards the realistic utilization of linked data. In this study, we consider retrieval of temporal linked data information from the fact-centric perspective. In future work, we want to extend our system to the document-centric perspective. We assume that inclusion of both perspectives will generate more concise output.

Chapter 5

LiCord: A Machine Learning-based Keyword Segmenter

In this chapter, we introduce a framework that can be used to devise keywords from natural language query. We first describe about its introduction (Section 5.1). Then we start describing our proposal (Section 5.2). We show results of implementing our proposal through experimental results and discussion (Section 5.3). Later, we discuss on our contribution and give guideline how the proposed framework can be extended to devise keywords for BoTLRet and TLDRet, described in Chapter 3 and 4, respectively (Section 5.4). Finally, in Section 5.5 we summarize the Chapter.

5.1 Introduction

We describe a proposal that identifies important word segments out of a the natural language sentence, which intuitively bear main sense of the sentence and prerequisite to select keywords (as described task (i.) among the three tasks for a Linked Data information access framework in Section 1.3 and 3.1). For example, for a natural language sentence "Who is Barack Obama married to?", the important word segments could be "Barack Obama" and "married". Over a sentence, the proposal automatically identifies some word segments, and classifies whether the word segments represent main sense of the sentence. In linguistics, such important word segments are mentioned as Content Words (CWs), and usually belong to nouns, most verbs, adjectives, and adverbs and refer to some object, action, or characteristic. To identify CWs, we check some structural features that relate text segments into CWs. We devise the features over a large text corpus and apply machine learning-based classification that classifies the segments into CWs. The proposed framework only uses large text corpus and some training examples, apart from these, it does not require any language specific tool.

One use of CWs in this thesis can be that they can be used to generate keywords for BoTLRet and TLDRet (described in Chapter 3 and 4, respectively). To generate keywords for BoTLRet and TLDRet, we first need to identify the CWs, then we need to link CWs with dataset vocabularies (as described task (ii.) among the three tasks for a Linked Data information access framework in Section 1.3 and 3.1). In this Chapter we mainly describe about the proposed CWs segmenting framework called **L**anguage **i**ndependent **Co**ntent Wo**rd** Segmenter **LiCord**. Later, we briefly outline how CWs can be linked towards the dataset vocabularies.

5.1.1 Motivation

In text mining, keywords play diverse roles e.g., finding of (new) topic [37, 75], sentiment analysis [50, 56], summarization of document [68], automatic answering of questions [96] etc. However, devising keywords for a natural language query is a daunting task, because it requires to i). identify word segment, and ii). consider whether the segmented words are important part of the text i.e., Content Words. Content Words (CWs) are words that belongs to nouns, most verbs, adjectives, and adverbs that refer to some object, action, or characteristic. Usually CWs are the meaning holding part of a sentence and carry independent meaning. Moreover, CWs are also open-words i.e, new words can be introduced as CWs [112]. For example, for a natural language query "Who is Barack Obama married to?", the CWs could be "Barack Obama" and "married".

We are motivated to identify CWs that they can be used to devise the keywords for our proposed Linked Data information access frameworks BoTLRet and TLDRet. As we see in Chapter 3 and 4, both BoTLRet and TLDRet are keyword-based information access frameworks, and these keywords intuitively present a natural language query. Furthermore, keywords in BoTLRet and TLDRet need to be exactly representative towards datasets' vocabularies. Therefore, devising keywords for a natural language query in BoTLRet and TLDRet require two major tasks

- identification of CWs for the natural language query (as described task (i.) among the three tasks for a Linked Data information access framework in Section 1.3 and 3.1)
- linking of CWs towards the datasets' vocabularies (as described task (ii.) among the three tasks for a Linked Data information access framework in Section 1.3 and 3.1)

The CWs finding framework is a Language Independent framework. It is required to keep the proposed Linked Data information access frameworks free from language dependent tools, because Chapter 3 shows how performance of those tools directly affects information access performance. The proposed framework only uses large text corpus and some training examples. Apart from these, it does not require any language specific tool. We check some structural features that relate text segments into CWs. We devise the features over a large text corpus and apply machine learning-based classification that classifies the segments into CWs. In CWs finding we contribute

(1.) by identifying structural features of text that separates CWs from other parts of the text (described in Section 5.2.3),

(2.) by devising a supervised machine learning model that classifies segment of the text into CWs (described in Section 5.2.4), and

(3.) by developing and evaluating the system (described in Section 5.3).

The below we mainly describe about the proposed CWs finding framework, and its performance by showing some test results and discussion. Later, we briefly outline how CWs can be linked towards the dataset vocabularies.

5.2 Language independent Content Word Segmenter (LiCord)

In this section, we describe the proposed framework **LiCord** in detail. By LiCord, we automatically identifies some word segments, and classifies whether the word segments represent main sense of the sentence. Therefore, for some text segments, LiCord mainly generates a classification model which can classifies the text segments into CWs.

In LiCord, we use

(i.) *language Ļ specific large text corpus Ŧ* − usually the corpus should be large enough so that it covers quite high number of vocabularies and sentence structures

(ii.) *language Ļ specific training examples* − usually the examples should include various type of CWs such as nouns, named entities, verbs, adjectives, and adverbs over the Ŧ.

By observing training examples over the Ŧ, we identify some structural features and learn some machine learning classification model that classifies text segments of Ŧ into CWs.

Figure 5.1 shows work-flow of this classification model learning. It holds four processes:

1. NGram Constructor
2. Function Word Decider
3. Feature Value Calculator
4. Classifier Learner

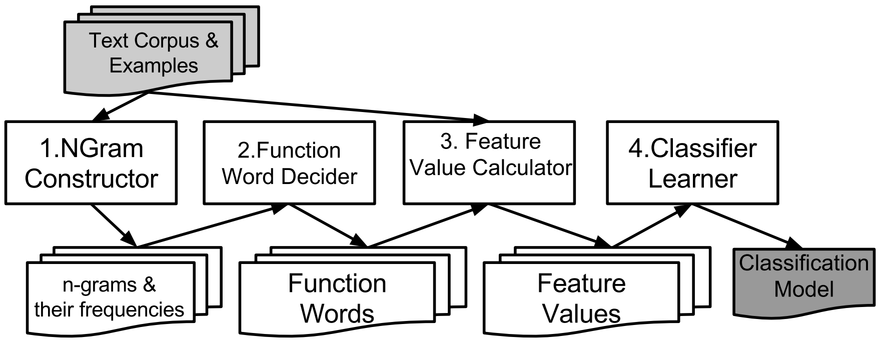


FIGURE 5.1: Work-flow of LiCord classification learning

TABLE 5.1: Variable length n-grams and their frequencies for an exemplary corpus Ŧ= "Japan is an Asian country. Japan is a peaceful country".

	n-grams and frequencies over the Ŧ
size 1 n-gram (/uni-gram)	{[Japan−2], [is−2], [an−1], ..., [country−2], [a−1], ... }
size 2 n-gram (/bi-gram)	{[Japan is−2], [is an−1], ..., [Asian country−1], ...}
size 3 n-gram (/tri-gram)	{[Japan is an−1], [is an Asian−1], [an Asian country−1], ... }

Here Process 1 performs text segmentation, Process 2 and 3 devise feature values for the segments, and Process 4 learns classification model to decide the segments into CWs. Below we describe each process in brief.

5.2.1 NGram Constructor

In NGram Constructor, we segment corpus Ŧby word token segmenting tag such as "space" (i.e., " ") and construct word n-grams[1] $\mathcal{N}$ between size 1 and η. Niesler et.al., mentioned such kind of n-grams as variable length n-grams [76]. In our description, we will use variable length n-grams as n-grams unless we want to discuss about their sizes. For each n-gram $x \in \mathcal{N}$ we also count its frequency $frq(x)$ over the Ŧ.

For example, Table 5.1 shows variable length n-gram of size between 1 and 3 for some exemplary corpus "Japan is an Asian country. Japan is a peaceful country". In the Table, the first column indicates variable length n-grams, and the second column shows n-grams and their frequencies. We show each n-gram and its frequency value in a square bracket (i.e., []) by putting a hyphen mark (i.e., "−") in between.

[1]in later part, we will use n-gram(s) to mean word n-gram(s)

Algorithm 1: Selection of non-English FWs

```
Input: English Function Word set $EFWs$, n-grams $\mathcal{N}$, threshold $\alpha$
Output: non-English FWs $nEFWs$
$X_\alpha \leftarrow$ TopFrequent($\mathcal{N}, \alpha$) ;
$nEFWs \leftarrow \emptyset$ ;
foreach $x \in X_\alpha$ do
    $et \leftarrow$ EnglishTranslation($x$) ;
    $tokenSet \leftarrow$ Tokens($et$) ;
    if $(EFWs \cap tokenSet) \neq \emptyset$ then
        $nEFWs \leftarrow \{nEFWs \cup x\}$
return $nEFWs$
```

5.2.2 Function Word Decider

In Function Word Decider, we decide language Ļ specific Function Words (FWs). Usually FWs indicate important sentence structural morphology [57]. We use them to harness these morphologies.

By definition, FWs are words that have little lexical meaning or have ambiguous meaning. In a sentence, FWs serve to express grammatical relationships with other words, or specify the attitude or mood of the speaker [57].

For English language, the FWs are well documented. Therefore, we use them directly (collected from Bunkyo Gakuin University site[2]). For a non-English language, we decide them heuristically. The basic idea of this heuristic is that usually the FWs are frequently appeared words, therefore we will search them in frequent n-grams.

To decide non-English FWs, we first pick threshold α number of top frequent n-grams from $\mathcal{N}$. Then for the top frequent α number of n-grams, we translate them into English and filter them for FWs. Algorithm 1 describe identification of FWs for a non-English language. Here function $TopFrequent(\mathcal{N}, \alpha)$ picks α number of top frequent n-grams from $\mathcal{N}$. For each n-gram $x \in X_\alpha$, function $EnglishTranslat\ ion(x)$ translates x into English and produces English translation et. Then function $Tokens(et)$ tokenize the English translation et and produces set of tokens $tokenSet$. We decide a n-gram x is a FW if any token of $tokenSet$ matches with English FWs $EFWs$. The decided FWs are returned as non-English FWs $nEFWs$. However, if an English Translation service (such as Google Trans-

[2]http://www2.fs.u-bunkyo.ac.jp/˜gilner/wordlists.html#functionwords

lator[3]) is not available for a non-English language, we decide all the top frequent α number of n-grams as FWs.

Below we show selection of two Indonesian FWs. In Indonesian language, "yang" (="that") and "bagin dari" (="part of") are considered as two different FWs because both of them are found as first $\alpha = 1000$ frequent n-grams and some token(s) of their English translations matches with English FWs – "that" and "of" respectively.

Usually FWs are close-set words [112], therefore, unlike CWs, we need to decide FWs once only.

5.2.3 Feature Value Calculator

In Feature Value Calculator, we devise 15 different features for n-grams. We create the features by observing sentence structural morphology. The observation factors are as follow – i.) how frequently CWs are appeared in the whole corpus and the croups sentences [3], ii.) where CWs usually are appeared over the corpus sentences i.e., at the beginning/middle/last part of sentences [3, 68], iii.) how CWs relate other words such as FWs [57], and iv.) how CWs follow punctuation marks (e.g., ",", "–", etc.) [3].

Table 5.2 shows features and their calculations. Here the first column shows features with their number. The second column describes how we calculate each feature value. We calculate feature values over the corpus Ŧ and sentences Ş of Ŧ. We assume that sentences will hold language specific period marks such as, "."(dot), or "|"etc. Here x is a n-gram.

We used the above features and their calculation techniques in next process and collect feature values which are used to learn classification model.

5.2.4 Classifier Learner

In Classifier Learner, we collect feature values for $\mathcal{N}$ (n-grams). Then we use some classification algorithm and learn classification model that predicts whether a n-gram should be CW.

[3]https://translate.google.com/

Table 5.2: Features and their calculations

	Feature Calculation
f_1	number of tokens in x.
f_2	frequency of x over the Ŧ.
f_3	number of sentences Ş that contains x.
f_4	number of sentences Ş that starts with x.
f_5	number of sentences Ş that ends with x.
f_6	number of sentences Ş that contains x but not at the start and at the end.
f_7	number of FWs contained by x.
f_8	Boolean value whether the first token of x is a function word (FW) f.
f_9	Boolean value whether the last token of x is a FW f.
f_{10}	number of sentences Ş that simultaneously contain n-gram x and a FW f in a way that f always appears immediate left to the x.
f_{11}	number of sentences Ş that simultaneously contain n-gram x and a FW f in a way that f always appears immediate right to the x.
f_{12}	Boolean value whether the first token of x is a training example.
f_{13}	Boolean value whether the right token of x is a training example. For one token x, first and last token are same.
f_{14}	number of sentences that simultaneously contain n-gram x and a punctuation mark p in a way that p always appears immediate left to the x.
f_{15}	number of sentences that simultaneously contain n-gram x and a punctuation mark p in a way that p always appears immediate right to the x.

However, since n-grams are huge in number, feature value calculation for them is not scalable. Moreover, randomly picking of n-grams is not enough to represent the whole corpus. Usually in the classification model learning, the whole corpus representing data is a requirement. We handle the scalability issue by dividing $\mathcal{N}$ into several n-gram sets which represent the whole corpus. Each of these sets hold n-grams with similar range frequencies. For similar range n-gram frequencies, we select some randomly picked n-grams and then devise a classification model for that range. This frequency based approach works because n-grams with similar kind of frequency values should hold similar kind of feature values.

To get frequency range based n-grams, we group n-grams for their frequency range values. If a frequency range value is i to j (i, j), we get n-grams $X_{(i,j)}$ as below

$$X_{(i,j)} = \quad \{x \mid x \in \mathcal{N} \wedge i \leq frq(x) \leq j\}$$

Here x is an n-gram. We consider that no two n-grams fall into two different sets. Therefore, if two range values are (i, j) and (k, l), we constrain range values as $i \leq j < k \leq l$.

Algorithm 2: Selection of frequency range based training n-grams

```
Input: Frequency range based n-grams $X_{(i,j)}$, maximum size of n-gram $\eta$,
       threshold $\beta$
Output: Frequency range based training n-grams $T_{(i,j)}$
$S_{pos} \leftarrow \{x \mid x \in X_{(i,j)} \wedge \boldsymbol{isPos(x)} = true\}$ ;
$S_{neg} \leftarrow \{x \mid x \in X_{(i,j)} \wedge \boldsymbol{isPos(x)} = false\}$ ;
$T_{(i,j)} \leftarrow \emptyset$ ;
for $t \leftarrow 1$ to $\eta$ do
    $tmp_{pos} \leftarrow \{x \mid x \in S_{pos} \wedge \boldsymbol{nToken(x)} = t\}$ ;
    $tmp_{neg} \leftarrow \{x \mid x \in S_{neg} \wedge \boldsymbol{nToken(x)} = t\}$ ;
    $T_{(i,j)} \leftarrow \{T_{(i,j)} \cup \boldsymbol{selRandom(}tmp_{pos}, \beta\boldsymbol{)}\}$ ;
    $T_{(i,j)} \leftarrow \{T_{(i,j)} \cup \boldsymbol{selRandom(}tmp_{neg}, \beta\boldsymbol{)}\}$ ;
return $T_{(i,j)}$
```

Even these frequency range based n-gram sets hold huge number of n-grams, therefore within a set, we pick them randomly. To collect training n-grams from a $X_{(i,j)}$ set, we select them for their each n-gram size positive examples and each n-gram size negative examples. Algorithm 2 shows this selection. It takes the frequency range based n-grams $X_{(i,j)}$, maximum size of n-gram η and threshold β and generates the frequency range based training n-grams $T_{(i,j)}$. Here we use three functions: $isPos(x)$, $nToken(x)$ and $selRandom(tmp, \beta)$. The function $isPos(x)$ identifies whether x belong to the example, the function $nToken(x)$ identifies number of token in x and the function $selRandom(tmp, \beta)$ randomly selects β number of elements from the set *tmp*. The main idea of this Algorithm is that we separate $X_{(i,j)}$ into positive and negative sets. Then, for each n-gram size 1 to η, we randomly select β number of positive and β negative examples. This kind of training set construction reduces classification bias because the training examples are representative for each token, and each kind of examples.

So, this frequency range based feature values are used to learn different classification models which later are used to classify n-grams for their respective frequency range based model. In Classifier Learner, we learn individual classification model $CM_{(i,j)}$ for frequency range based n-grams $X_{(i,j)}$. We use binary class classification algorithm to learn the classification model.

5.3 Experiment

LiCord is a Language Independent framework. It is required to keep the proposed

Table 5.3: Statistics of corpus and corresponding n-grams

Language	**Size of Corpus**	**# of n-gram**
English (1/20th part of the corpus)	500 MB	170M
Vietnamese	500 MB	108M
Indonesian	300 MB	29M

Linked Data information access frameworks free from language dependent tools, because Chapter 3 shows how performance of those tools directly affects information access performance. Therefore, we experimented LiCord on three different languages: i.) English ii.) Vietnamese iii.) Indonesian to show that LiCord can find CWs in Language Independent way.

To train the classification model, we used language Ļ specific Wikipedia page contents Ŧ as large text corpus. We find approximately 35M (1M = 10^6), 2M, 1.5M, different Wikipedia pages for English, Vietnamese, and Indonesian languages respectively. On the other hand, we use page titles of those Wikipedia pages as training examples. We consider page titles as training examples because they, by large, describe the CWs which belong to nouns, or noun phrases, or named entities, verbs, etc.

In experiments, we set $\eta = 5$. To decide FWs (function words) for non-English languages, we set $\alpha = 1000$. Moreover, to select training n-grams, we set $\beta = 2000$. Over the corpus, we used SRI Language Modeling Toolkit[4] to calculate variable length n-grams and their n-gram frequencies. Table 5.3 shows statistics of different language corpus and their corresponding number of variable length n-grams of size between 1 and 5.

We learned classification model for n-grams with different n-gram frequency ranges such as (1,1), (2,2), (3,4), (5,9), (10,14), (15,19) etc. For each model, we collected feature values for randomly picked 20,000 training n-grams: for each token (i.e., 1 to 5) 2000 as positive and 2000 as negative examples. In preliminary experiment, we learned classification model for two binary classification algorithms: C4.5 [82] and SVM [28] and found C4.5 performs better than SVM. Therefore, in LiCord, we decided to use C4.5-based classification model.

We experimented our proposed framework LiCord to serve two major purposes. First, we checked LiCord that whether it can identify NEs (Named Entities), and

[4]http://www.speech.sri.com/projects/srilm/

parser like parts of speeches. Since contemporary researches [55, 68, 73] used NE annotation tool to find CWs, LiCord also should find the NEs. On the other hand, LiCord identified CWs should hold nouns, verbs, adjectives, and adverbs which should be varified by existing parser. Second, we checked that whether LiCord can identify CWs in a language independent way. We also checked whether LiCord identified CWs follow open-words property

5.3.1 Experiment 1

In first experiment, we checked LiCord for identification of NEs and parts of speeches. The related researches used NE annotation tool to find CWs, therefore LiCord also should find the NEs. On the other hand, parser generated nouns, verbs, adjectives, and adverbs should be included in LiCord identified CWs, therefore we verify them to know their coverage.

5.3.1.1 The Named Entities (NEs)

To check whether LiCord can identify NEs, we annotated some text using LiCord and compare the annotation outputs with two NE annotators. Usually an annotator identifies some part of the text into some defined set of items. We compared LiCord with two Wikipedia title annotators: Wikifier [90] and Spotlight[5] [73]. Both the annotators identify text to Wikipedia titles. Since most of the cases the Wikipedia titles are NEs, we assume that Wikipedia annotation corresponds the NE annotation.

For some given text, LiCord can check whether some of its n-grams or text segments should be positively classified. Therefore, if Wikipedia titles are available over the text, LiCord should identify them as the positively classified n-grams. Thereby, LiCord can be considered as Wikipedia title annotator. However, this comparison can be done for English language only because the other two annotators do not support languages like Indonesian, Vietnamese.

[5]more accurately DBpedia annotator, DBpedia works as structured version of Wikipedia, it can be found at http://dbpedia.org/about/

Table 5.4: Comparison for LiCord with Wikifier

	Recall
Wikifier	33.33%
LiCord	90.47%

Table 5.5: Comparison for LiCord with Spotlight

	Recall
Spotlight	83.33%
LiCord	91.66%

We compared LiCord with Wikifier and Spotlight for the given text in the respective demo sites[6]. To compare LiCord with Wikifier, we used the Spotlight given demo text. Over the Spotlight given demo text, we first executed Spotlight itself which generated 21 Wikipedia titles. Then we executed Wikifier and LiCord for the the same text and measured how many Wikipedia titles are identified by both systems respectively. On the other hand, to compare LiCord with Spotlight, we used Wikifier given demo text and follow the likewise procedure with respective systems. Over the Wikifier demo given text, Wikifier generated 12 Wikipedia titles.

Table 5.4 and 5.5 show annotation comparisons in recall values for LiCord with Wikifier, and LiCord with Spotlight. It shows, LiCord identified more number of Wikipedia titles than the other two systems. Therefore, we consider that LiCord can identify the NEs. In LiCord, we checked a large number of sentence structural features over a big corpus and then classified the n-grams with their respective frequency range based models which identified more number of Wikipedia titles than the other two systems.

5.3.1.2 The Parts of Speech

To check whether LiCord can identify parser like parts of speeches, we executed LiCord and parser on some text. Usually a parser identifies parts of speech, we collected the nouns, verbs, adjectives and adverbs for the text. Since LiCord identified CWs should hold nouns, verbs, adjectives and adverbs for the text, we compared recall values between parser output for nouns, verbs, adjectives and adverbs, and the LiCord identified CWs. Since English has state-of-the-art parser such as Stanford Parser[7], we experimented this for English language only.

[6]http://cogcomp.cs.illinois.edu/page/demo_view/Wikifier (for the example of GoogleChina), and http://dbpediaspotlight.github.io/demo/, respectively

[7]http://nlp.stanford.edu/software/lex-parser.shtml

TABLE 5.6: Comparison for LiCord with Parser

Language	Recall
English	92.30%

We executed LiCord and Stanford parser on Spotlight demo given text. On Spotlight demo given text, Stanford parser generated 26 words as nouns, verbs, adjectives and adverbs, among which LiCord identified 24 of them. Table 5.6 shows comparison result with Stanford parser. Achieving of good recall informs that the devised structural features were effective which identified most of the required parts of speeches.

5.3.2 Experiment 2

In second experiment, we checked that whether LiCord can identify CWs in a language independent way. We also checked whether LiCord can maintain open-word property of CWs. Usually CWs are open-words that is new words can be introduced as CWs [112].

To check whether LiCord can identify CWs in a language independent way, we experimented LiCord over some test n-grams that belong to three different languages and checked their classification accuracy. If LiCord can classify both positive and negative test n-grams with high accuracy, we can consider that LiCord serves our research goal – the finding of CWs in a language independent way.

As discussed, the individual classification model $CM_{(i,j)}$ works over the n-grams that hold n-gram frequency between i and j. Here, the test n-grams are different from the training n-grams. For each n-gram frequency range (i, j), we randomly picked 2,000 test n-grams: 1,000 positive and 1,000 negative. In English, example of such test n-grams are "ahead of all" (size 3 n-gram), "Mayors of New York" (size 4 n-gram) etc. We executed LiCord classification model for each test n-gram. For a true classified n-gram, if it was found in Wikipedia title, we considered that the proposed framework worked correctly, otherwise it worked incorrectly. For a false classified n-gram, it works with the opposite manner.

In Table 5.7 we show classification accuracy (in %) over test n-grams. We present this result for n-grams with five different n-gram frequency ranges: (1,1), (2,2),(3,4),

TABLE 5.7: CW finding classification accuracy % in test

Frequency Range	English	Indonesian	Vietnamese
(1,1)	76.68	90.56	90.30
(2,2)	83.00	93.20	94.15
(3,4)	84.37	94.23	94.76
(5,9)	83.87	95.89	93.97
(10,14)	87.09	96.15	94.95
Average	**83.25**	**93.80**	**93.54**

(5,9) and (10,14). We selected those ranges to grasp the evaluation results for different ranges n-grams from small to large. Here, the first column shows n-grams' frequency range. The remaining columns show test n-gram classification accuracy (in %) for English, Indonesian and Vietnamese languages respectively. Here we find that while the classification accuracy for Indonesian and Vietnamese languages was high, the classification accuracy for English language was low. Our detail observation indicates that drop of accuracy in English could come from a reason that some *false* n-grams (i.e., not found in Wikipedia titles) were classified as *true* that is they were discovered as new CWs. For example, in English, LiCord discovered "Australian Air Force Military" as a new CW which was not appeared as Wikipedia title previously. Since Table 5.7 accuracy was calculated ignoring the newly discovered CWs, we examined the non-accurate classified test n-grams and checked whether they correctly identified as new CWs. To evaluate this discovery accuracy, we engage 3 native users for each languages and verified whether the discoveries were correct. We verified the result for majority voting. Table 5.8 shows newly discovered CWs finding accuracy (in %) for non-accurate classified test n-grams of Table 5.7. Here, the first column shows n-grams' frequency range for which the CWs were discovered, the remaining columns show discovery accuracy (in %) for English, Indonesian and Vietnamese languages respectively. It is seen that the discovery of CWs in English is more accurate (47.45%) than the other languages. In our observation, we found that since English language corpus is big and hold various sentence structural morphologies, the feature value calculation were more accurate which learned more accurate classification model.

We also found that the most of the cases, when n-gram frequencies are more, the classification accuracy are higher. Moreover, we also found that the classification accuracy gets increased, if the number of tokens in n-grams are more. For example, for n-gram frequency range 10-14 in English, we achieved accuracy 81.25%,

TABLE 5.8: Newly discovered CWs finding accuracy % for non-accurate classified test n-grams of Table 5.7

Frequency Range	**English**	**Indonesian**	**Vietnamese**
(1,1)	27.90	11.34	10.63
(2,2)	45.00	18.54	25.00
(3,4)	52.11	24.45	27.56
(5,9)	50.34	25.56	30.88
(10,14)	61.90	29.89	35.13
Average	**47.45**	**21.95**	**22.50**

85.75%, 91.49% and 89.50% for 2 tokens, 3 tokens, 4 tokens and 5 tokens n-grams respectively.

Anyway, as an initial evaluation, the overall accuracy was 83.25% for all three languages which we consider a reasonable performance. Therefore, we consider that LiCord can identify CWs in a language independent way. This performance comes because LiCord checked a large number of sentence structural features which correctly generated classification models and classified n-grams into Wikipedia titles (or CWs).

5.4 Vocabulary Specific Keyword Linker

The LiCord identified CWs are used to devise the keywords for BoTLRet and TLDRet that supports task (i.) as described in Section 1.3 and 3.1). Then we need to link CWs towards the datasets' vocabularies that supports task (ii.) as described in Section 1.3 and 3.1). To link the keywords, we can use state-of-the-art entity linking framework [38]. In [38], it uses a probabilistic approach that makes use of an effective graph-based model to perform collective entity disambiguation. Therefore linkable words between CWs and datasets' vocabularies are considered as keywords, that are disambiguated jointly across the dataset by combining a dataset-level prior of entity co-occurrences with local information captured from the natural language query, their corresponding CWs and their surrounding context. This state-of-the-art entity linking framework is based upon simple sufficient statistics extracted from the dataset. The experimental result shows the framework achieved 88% entity linking accuracy. Therefore, we can use it to find exact vocabularies from the dataset. Furthermore, the framework does not use any language specific tool, therefore we can use it BoTLRet and TLDRet.

5.5 Summary

CWs describe important parts of the text which have various usage over the text mining. In this Chapter, we propose a framework that identifies CWs for a natural language query. Here we outlined that how a large text corpus can be exploited to identify CWs: particularly in feature calculation and scalability handling. We also showed experimental results and their findings. Furthermore, we also outline how CWs can be exploited to devise keywords for BoTLRet and TLDRet by adapting a state-of-the-art entity linking framework. Thereby, we discuss that together with LiCord and the state-of-the-art entity linking framework how we can automatically devise keywords for a natural query.

Chapter 6

ALDErrD: An Information Assessment Framework over Linked Data

We first describe about its introduction (Section 6.1). Then we we describe the basic idea of our proposed framework (Section 6.2). Next, we describe the proposed ALDErrD in details (Section 6.3). We show results of implementing our proposal through experimental results and discussion (Section 6.4). Later, we explain the proposed framework that how it solved the problem that existed in the contemporary systems (Section 6.5) Finally, in Section 6.6 we summarize the Chapter.

6.1 Introduction

Linked Data currently hold a vast amount of knowledge. However, Linked Data suffer in data quality, and this poor data quality brings the need to identify erroneous data. Because manual erroneous data checking is impractical, automatic erroneous data detection is necessary.

According to the data publishing guidelines of Linked Data, data should use (already defined) ontology which populates type-annotated Linked Data. Usually, the data type annotation helps in understanding the data. However, in our observation, the data type annotation could be used to identify erroneous data. Therefore, to automatically identify possible erroneous data over the type-annotated Linked Data, we propose a framework that uses a novel nearest-neighbor based error detection technique. We conduct experiments of our framework on DBpedia, a type-annotated Linked Data dataset, and found that our framework shows better performance of error detection in comparison with state-of-the-art framework.

6.1.1 Motivation

Usually, Linked Data are generated either manually or automatically. However, both generation procedures have some flaw which produce erroneous Linked Data entries. Usually the manual intervention-based procedure generates more cleaner data, however, such data also contain erroneous entries because of human errors. Moreover, such data could be generated from multiple sources which sometime differ one another. Example of such data PubMed, DrugBank etc. On the other hand, the automatic procedure generates more data than the manual intervention-based procedure because of easy and automatic procedure. However, such data are more prone towards erroneous data gathering. One reason of such erroneous data gathering is the wrong contents in main data source, from which Linked Data are automatically extracted. If the main data source contain wrong entries, the generated Linked Data also extract wrong entries. The example of such wrong entries could be wrong entries of DBpedia that are extracted because of wrong Wikipedia contents. Moreover, erroneous entries could also be extracted because of problematic data extraction. Therefore, according to Linked Data generation procedures, they are potential to contain errors [58, 87]. However, to use such

Linked Data effectively, data consumers commonly expect to easily retrieve high-quality data. This brings the need to identify erroneous data in the Linked Data. Usually, manual erroneous data checking is impractical. Therefore, automatic erroneous data detection is necessary.

On the other hand, according to the best-practice data publishing guidelines [47] of Linked Data, data should use (already defined) ontology which populates type-annotated Linked Data (See Chapter 2 in Section 2.1.3 and Linked Data part of in Figure 1.1). In the Figure, the Linked Data part hold type annotation for instance "rc:cygri" using "rdf:type" as "Person". In real world, DBpedia is a type-annotated Linked Data dataset. Usually, the data type annotation helps in understanding the data. However, in our observation, the data type annotation could be used to identify erroneous data. The intuition behind this assumption is that the same type of Linked Data resources should share the same kind of values. Therefore, if data values of some Linked Data go beyond the usual pattern or trend of other same type of Linked Data, we consider them as erroneous data. The assumption fully comply the data outlier detection, which is a common in erroneous data identification. Chapter 2 has already outlined some common outlier detection methods, we will utilize some of them in current quality assessment. However, it is worth mentioning that this outlier-based error detection might not be always true, but it gives opportunity to check the data to find erroneous data over the type-annotated Linked Data.

In the past, some studies have dealt with erroneous data findings in the Linked Data. However, these studies have their own limitations. For example, some require Linked Data domain-level expertise [1, 59, 116]. Some require another similar data source [19, 58, 74], are not suitable for diverse datasets, or are impractical for large datasets. Other works are for specific data types and ignore the errors for the remaining data types [34, 111].

In this study, we are motivated to investigate the above mentioned issues.

The previous two chapters (Chapter 3 and 4) have dealt with Linked Data information access, which is a big issue in Linked Data success. In this Chapter, we will investigate another crucial Linked Data issue – Linked Data quality assessment – that can make Linked Data a success [8]. We understand that users will be interested to access the assessed data, so that they can rely upon on them.

6.1.2 Contributions

We propose a framework to identify possible candidate of erroneous data over the type-annotated Linked Data. The framework is named **A**uto **L**inked **D**ata **Err**or **D**etector **ALDErrD** [87] which automatically detect potential error patterns and predict possible candidate of erroneous data. The main features of our proposed framework ALDErrD are the following:

(I.) It is free from manual intervention

(II.) It does not require domain-level expertise

(III.) It does not require other data sources of the same kind

(IV.) It is suitable for any type of data

6.2 Basic Idea

Here we will describe the basic idea of our research framework. To do this, first we will exemplify the type-annotated Linked Data, then we will share the idea.

As mentioned, the best-practice Linked Data data publishing recommend to use (already defined) ontology, and it populates type-annotated Linked Data. For example, if there are two RDF triples <res:[1]Michael_Jordan, ont:[2]height, 168.0 >[3] and <res: Michael_Jordan, rdf:[4]type, ont:BasketballPlayer>, the latter one typify the res:Michael_Jordan Linked Data resource as "Basketball Player". Usually, the type annotation generalizes the Linked Data resources. We use this generalization to find candidates of erroneous data over the Linked Data.

To identify candidates of erroneous data in the Linked Data, we assume that the same type of Linked Data resources share commonalities. In particular, we assume that the same type of Linked Data resources share the similar kind of values for the same Property[5]. For example, in an Linked Data dataset, if there are good number of Linked Data resources are typified as "Basketball Player" (i.e.,

[1]http://dbpedia.org/resource/
[2]http://dbpedia.org/onto/
[3]Throughout the paper, examples are shown from DBpedia 3.8
[4]http://www.w3.org/1999/02/22-rdf-syntax-ns#
[5]Property can be inter-changed by Predicates in the RDF triple

ont:BasketballPlayer), and the resources also hold "height" (i.e., ont:height) values, we should expect the height values would be similar kind of values. Therefore, for resources of a particular type Linked Data, if literal values go beyond the usual pattern or trend of other resources of the same type of Linked Data, we consider them as candidates of erroneous data. This idea is generally rational. For example, we expect individuals who are Basketball Players to be taller. So, if an individual Basketball player is not as taller as the most of the Basketball players, we can predict that the data might be wrong. However, the above assumption might not always be true, but it gives the option to check the data.

Technically, the above assumption has also been well studied in unsupervised outlier detection and is called nearest-neighbor based error detection [21]. In such a case, it is assumed that normal data instances occur in dense neighborhoods, while errors occur far from their closest neighbors [16, 17]. So, error detection requires a similarity/distance measurement defined between/among the data [115]. In the type-annotated Linked Data, nearest-neighbor based error detection is well suited for the variant called "multivariate nearest-neighbor based error detection" [99], because such error detection depends upon the attributes of data and, usually, the type-annotated Linked Data hold several such attributes (e.g., type, domain, range, etc.).

On the other hand, since the Linked Data are generated from various sources, keeping conformity among the data is a challenge. The presence or absence of a particular attribute of data or using data values in different formats might present the same kind of data in different ways. Usually, the ontology of the Linked Data would restrict such varieties. However, in a real-world scenario, adhering to a strict ontology in the Linked Data is not feasible. It introduces the requirement of grouping data instances for the presence or absence of attributes and the formatting of data values. For grouped data, it is assumed that normal instances lie close to their closest group centroid, whereas erroneous instances lie far away from their closest group centroid [49, 89, 97, 99]. Therefore, we adapt the nearest-neighbor based error detection for groups.

6.3 Framework Details

In this section, we describe our proposed framework ALDErrD in detail. We take a Class (such as Person) and a Property (such as Birth Date)[6] as input, and detect whether the Linked Data resources hold erroneous literal values for the Objects of the given Property. Usually, a Class information typify an Linked Data resource. Preferably, to check some Linked Data resources for their erroneous literal values of Object, we should select them for their most specific Class. This is because, an Linked Data resource can be typified by multiple Classes, but the most specific Class will define it more precisely. The rdfs:subClassOf closure is used to determine the most specific Classes for Linked Data resources. However, whether Linked Data resources belong to the most specific Class or some other super Classes, ALDErrD works for any given Class. But as mentioned, the most specific Class will identify candidates of erroneous data more accurately.

In ALDErrD, the detected errors are for erroneous Object values. We consider the detected errors as "Type-1 Errors". Over an Linked Data dataset, the Type-1 Errors appear because of

- Erroneous Content – data with wrong values (e.g., wrong actual values), and
- Erroneous Syntax – data with wrong syntactic patterns (e.g., wrong value format, wrong string pattern etc).

However, Type-1 Errors can be originated for wrong Linked Data attributes (e.g., type, domain, ontology etc). We consider such errors as "Type-2 Errors" and classify them into four kinds:

i. Erroneous Type – data with wrong Type attachment towards the Linked Data resources.

ii. Erroneous Domain – data with wrong Domain attachment towards the Linked Data resources.

iii. Erroneous Range – data with wrong Range attachment towards the Linked Data resources.

[6] In DBpedia, Person is http://dbpedia.org/Ontology/Person and Birth Date is http://dbpedia.org/property/birthDate

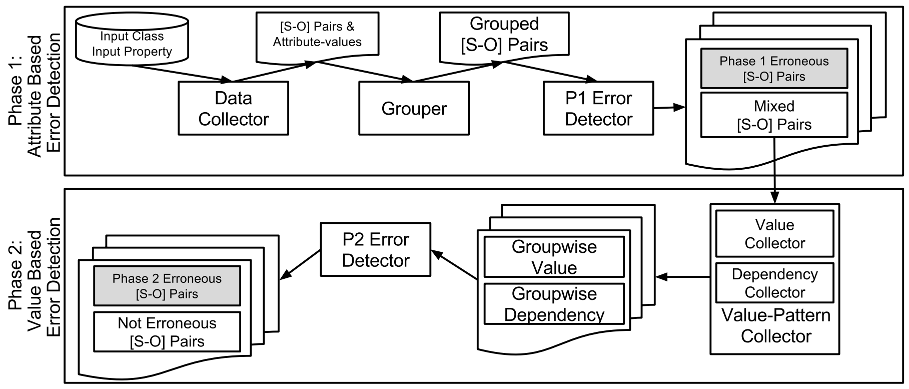

Phase 1: Attribute Based Error Detection
Input Class Input Property
[S-O] Pairs & Attribute-values
Grouped [S-O] Pairs
Data Collector
Grouper
P1 Error Detector
Phase 1 Erroneous [S-O] Pairs
Mixed [S-O] Pairs
Phase 2: Value Based Error Detection
Phase 2 Erroneous [S-O] Pairs
Not Erroneous [S-O] Pairs
P2 Error Detector
Groupwise Value
Groupwise Dependency
Value Collector
Dependency Collector
Value-Pattern Collector

Figure 6.1: Work-flow of Proposed Framework ALDErrD

iv. Erroneous Property – data with wrong Property attachment towards the Linked Data resources.

Therefore, if the Type 2 errors are identified, they further reveal the causes behind the Type 1 errors. By ALDErrD, we only detect the Type-1 errors, and if we want to classify them into Type-2 errors, we need to investigate them for further analysis. Moreover, since Type-1 and Type-2 Errors may co-exist for same [S-O] pairs, clear distinction between them is not always possible.

Figure 6.1 shows the work-flow of ALDErrD. We divide the proposed framework into two phases: Phase 1 – Attribute Based Error Detection and Phase 2 – Value Based Error Detection. In Attribute Based Error Detection, we group data for some attribute values. Such groups help in detecting a Phase 1 data error. In Value Based Error Detection, we take Phase 1 data that are still not considered as errors. Here we investigate data values and apply various nearest-neighbor based error detection techniques to identify possible anomalies. In Phase 1, we introduce a technique to group Linked Data resources which leads later steps of the framework, therefore we implement Phase 1 before the Phase 2. It also reduces the execution time of Phase 2 because, in such a work-flow, the Phase 2 only requires to identify errors over the filtered-out data of Phase 1. Below we describe both phases in detail.

6.3.1 Attribute Based Error Detection

The upper half of Figure 6.1 shows the work-flow of Attribute Based Error Detection. For the given input (i.e., a Class and a Property), first we use a process called Data Collector and collect Subject-Object ([S-O]) pairs (described below) along with some attribute values. Then, according to the attribute values, we use a process called Grouper and find groups among the [S-O] pairs. Then, for the grouped [S-O] pairs, we use a process called the P1 Error Detector and detect possible erroneous [S-O] pairs. Here, an [S-O] pair is erroneous data, if literal value of O of [S-O] is not correct.

In Phase 1, we utilize attributes as the main indicators to detect erroneous [S-O] pairs. Below we describe each process in detail.

6.3.1.1 Data Collector

We collect Linked Data resources for all RDF triples <Subject, rdf:type, Class>. Then, for each Subject, we collect [S-O] pairs for RDF triples <Subject, Property, Object>, where "S" represents Subject and "O" represents Object. Apart from [S-O] pairs, we also collect five different attribute values for each S and each O of [S-O] pairs. The attributes are i) type of literal value (LVT), ii) associated properties (PRT), iii) associated classes (CLS), iv) associated domain (DOM), and v) associated range (RNG). Practically, the literal value of Object will be used to identify the error. To do so, the LVT information largely allows us whether literal values are holding same kind value, so we collect the LVT. On the other hand, the remaining four attributes (i.e., PRT, CLS, DOM and RNG) will be used to check whether the same type Linked Data resources uses same attributes. We use the above-mentioned attributes because they possibly can be found in a type-annotated Linked Data. However, the readers can include further attributes that might produce better result. But in current ALDErrD setting, we use the above-mentioned attributes.

The finding of the LVT requires some processing, whereas finding the values of the other four attributes (i.e., PRT, CLS, DOM and RNG) just require data picking. Below we describe the value collection of these two types of attributes.

- **LVT**. We first find the literal value, and then find the type of literal value (LVT). For S of an [S-O] pair, it always holds the URI, while O always holds either the URI or a literal value. If S or O holds a URI, we extract its label for treating it as a literal value. However, it is possible that the Linked Data does not contain the label of O or the label of S. In such a case, we consider the URI as a literal value.

If O of an [S-O] pair is a literal value, usually the value is annotated by the data type. We consider this data type as a schema-defined type. For [S-O] pairs, we collect unique schema-defined types (SDTs) for literal values of O. However, it is not guaranteed that the literal value will always hold the data type annotation. In such a case, we need to devise the LVT. For any literal value that does not hold the data type annotation, we classify their LVTs into four types: STRING, DATE/TIME, NUMBER or URI. We adapt this classification from the study [118]. If we find that the literal value is only a URI, we consider the LVT as a URI. Otherwise, we execute a language parser over the literal value and determine the LVT from the named entities (NEs) of the parsed output. The following equation gives us the LVT, where "x" is either S or O.

$$\text{LVT(x)} = \begin{cases} \text{data type} & \text{(if data type is defined)} \\ \text{URI} & \text{(if literal value is URI only)} \\ \text{DATE/TIME} & \text{(if parsed output of literal value} \\ & \text{hold DATE/TIME type NE)} \\ \text{NUMBER} & \text{(if parsed output of literal value} \\ & \text{hold NUMBER type NE)} \\ \text{STRING} & \text{(otherwise)} \end{cases}$$

For example, we might have the [S-O] pair [rsc:Tom_Cruiz $-$170.18$\wedge\wedge$[7]centimeter], where 170.18 is the literal value annotated by using symbol ($\wedge\wedge$) centimeter, and it is the LVT(O). Another exemplary [S-O] pair could be [rsc:Tom_Cruiz$-$1962-07-03], where 1962-07-03 is a literal value, but it does not have the data type annotation, so the language parser identifies it as DATE/TIME.

- **PRT, CLS, DOM and RNG**. We use RDF triple patterns to collect the attribute values. The values of PRT and CLS can be found if S or O of the [S-O] pair is a URI. On the other hand, the values of DOM and RNG can be found when PRT exists. Therefore, if S or O is not a URI and the required triple pattern does not exist over the Linked Data, we consider the respective attribute values as null. The below equations collect the PRT, CLS, DOM and RNG values, respectively. In these equations, "x" is either

[7]http://dbpedia.org/datatype/

Table 6.1: Examples of attribute values for different Ss (or Os) for the Linked Data resource res:Tom_Cruise.

Attribute Name	**Attribute Value**
PRT(x)	ont:placeOfBirth, ont:dateOfBirth, ..., etc
CLS(x)	ont:Artist, ont:Actor,..., etc.
DOM(x)	ont:Person, ..., etc.
RNG(x)	ont:Place,rdfs:date..., etc

S or O. To extract CLS, DOM and RNG, we use 3 common properties type, rdfs:[8]domain and rdfs:range respectively.

$$\text{PRT(x)} = \begin{cases} \{?p \mid <x,?p,?o>\} & (\text{if x is URL } \wedge \\ & \exists <x,?p,?o>) \\ \text{null} & (\text{otherwise}) \end{cases}$$

$$\text{CLS(x)} = \begin{cases} \{?c \mid <x,type,?c>\} & (\text{if x is URL } \wedge \\ & \exists <x,type,?c>) \\ \text{null} & (\text{otherwise}) \end{cases}$$

$$\text{DOM(x)} = \begin{cases} \{?d \mid \forall p \in <x,p,?o>, & (\text{if } \exists p \in \text{PRT(x)} \wedge \\ <p,rdfs:domain, & \exists <p,rdfs:domain, \\ ?d>\} & ?d>) \\ \text{null} & (\text{otherwise}) \end{cases}$$

$$\text{RNG(x)} = \begin{cases} \{?r \mid \forall p \in <x,p,?o>, & (\text{if } \exists p \in \text{PRT(x)} \wedge \\ <p,rdfs:range,?r>\} & \exists <p,rdfs:range,?r>) \\ \text{null} & (\text{otherwise}) \end{cases}$$

Table 6.1 shows examples of the four attributes for different Ss (or Os). Here, the 1st column shows the attribute name, the 2nd column shows the attribute value.

All the collected information is further used to identify the possible erroneous [S-O] pairs.

[8]http://www.w3.org/2000/01/rdf-schema#

TABLE 6.2: Exemplary [S-O] pair groups divided by the horizontal double lines in the table.

[S-O] Pair	**LV(O)**	**LVT(O)**
[rsc:Gabriella_Hall−–09-05]	–09-05	MonthDay
[rsc:Armand_Dorian−6.0]	6.0	foot
[rsc:Nora_Danish−160.0]	160.0	centimeter
[rsc:Belinda_Hamnett−165.1]	165.1	centimeter
[rsc:Tom_Cruise−170.18]	170.18	centimeter
[rsc:MC_Jin−168.0]	168.0	centimeter
[rsc:Tsuchida_Bakusen−217.7]	217.7	centimeter
[...]	...	centimeter

6.3.1.2 Grouper

We group [S-O] pairs by the LVT(O). These groups help in predicting a data error.

We apply all grouping options for the LVT(O). For example, we group [S-O] pairs for the LVT(O) either as STRING, DATE/TIME, or data type. Table 6.2 shows examples of three such groups. Here [S-O] pairs are considered for Class "Artist" and Property "height". The columns show the [S-O] pair, the literal value of O (LV(O)), and LVT(O).

By the group-based erroneous [S-O] pairs identification approach, we try to understand semantic values among the groups. Understanding the semantic values is required to reduce identifying false positive errors. This is because, in Linked Data dataset some values could be syntactically very different but still they could be semantically similar. For example, in a Linked Data dataset, some person's "height" can be stored in centimeter and some are in inch, but there will be huge difference if we directly compare both group values. If the error detection framework identify errors for all [S-O] pairs together, it can identify false positive errors. Therefore, we individually treat each group and compare values for their semantics and identify erroneous [S-O] pairs.

Next, within the groups, we detect the possible erroneous [S-O] pairs.

6.3.1.3 P1 Error Detector

The Property Range and the attribute values from the previous process are used in detecting Phase 1 error candidates. The P1 Error Detector detects those error candidates. It uses three types of methods − (1) Property Range validation, (2) Group-wise [S-O] pair observation, and (3) [S-O] pair similarity score calculation. Below we describe them in details.

1. Usually, the Object of an RDF triple follows a constraint that the Object follows the Property Range. So, the Property Range helps in detecting erroneous [S-O] pairs. We collect it from the range value of an RDF triple pattern as <Property, rdfs:range, ?r>. In any case, if ?r is not found, we assign it. To do this, if a Property holds a token word "DATE", we assign it as DATE/TIME. For example, for prp:[9]birthDate, the Property Range would be DATE/TIME, because it includes the token word "Date". Otherwise, we consider the Property Range can hold any literal value types such as NUMBER, STRING, URI, SDTs. So, we detect potential erroneous [S-O] pairs by observing the Property Range and LVT(O). If LVT(O) does not belong to the Property Range, we consider such [S-O] pairs as potential candidate of erroneous [S-O] pairs.

2. After detecting the above candidates, we detect further candidates of erroneous [S-O] pairs by the number of pairs each group holds. Here, we check an [S-O] pair group to determine whether it could entirely or partially holds erroneous pairs. For a group, if the ratio between its number of [S-O] pairs in the group and the total [S-O] pairs of all groups is less than a threshold α, we consider the group as an erroneous group and detect its [S-O] pairs as error candidates. Table 6.2 shows three such groups among which the first two groups (i.e., rows) as error candidate examples.

3. Next, we try to find possible erroneous [S-O] pairs for groups that do not entirely hold erroneous [S-O] pairs. In such a case, we try to find possible erroneous [S-O] pairs for one group at a time. In a group G, we calculate each [S-O] pair's similarity score (simScore([S-O])) towards the other [S-O] pairs of G.

   To do this, we calculate similarity for each attribute of x, where x is either S or O. They are sim$_{PRT}$(x), sim$_{CLS}$(x), sim$_{DOM}$(x), and sim$_{RNG}$(x). We do not calculate similarity for LVT(x), because it was already considered when we made the groups.

[9]http://dbpedia.org/property/

To calculate similarity for an attribute ATT (= PRT, CLS, DOM or RNG) of x (sim_{ATT}(x)), we first accumulate the group attribute values GAV_{ATT}(x) as

$$\text{GAV}_{ATT}(\text{x}) = \begin{cases} \{\text{ATT(S)}|\forall\text{[S-O]}\in\text{G,[x-O]}\in\text{G}\} & \text{(if x is S)} \\ \{\text{ATT(O)}|\forall\text{[S-O]}\in\text{G,[S-x]}\in\text{G}\} & \text{(if x is O)} \end{cases}$$

Then, sim_{ATT}(x) is measured by |ATT(x)|/ |GAV_{ATT}(x)|.

So, for group G, we calculate each [S-O] pair's similarity score as

$$\text{simScore([S-O])} = \sum_{ATT\in\{PRT,CLS,DOM,RNG\},x\in\{S,O\}} \text{sim}_{ATT}(\text{x}).$$

In this way, we find similarity scores for all [S-O] pairs of group G. We detect possible erroneous [S-O] pairs of group G by finding the Outlier simScore([S-O]).

We calculate the Outlier based on the Interquartile Range (IQR) [108]. As we know, in the data error detection, an Outlier is a data value that resides far from other values, and the IQR is simple but effective way to identify such an Outlier. Here, for a rank-ordered data value set, quartiles divide them into four equal parts. The values that divide each part are called the first (Q1), second (Q2), and third (Q3) quartiles respectively. The Outlier point is below Q1 or above Q3 due to the consideration that it is measured by a factor of the IQR, and where the IQR itself is IQR = $Q_3 - Q_1$.

In our case, we consider the factor of IQR is 1.5. We adapt this factor from the research of Kontokostas et al [111] .

Data value smaller than Q_1 − 1.5*IQR and larger than Q_3 + 1.5*IQR is considered Outlier. However, the factor of the IQR could be varied. So, Outlier simScore([S-O]) holding the [S-O] pair is considered as a candidate of erroneous [S-O] pair.

The above described error candidate identification procedure is quite different from the basic technique. Over the Linked Data, the basic error candidate identification relies on property Range and other Linked Data Attribute values [74]. Usually, such property Range checking depends on a one particular Range value for an input property. However, in real-world scenario data are stored for different Ranges e.g., "height" of person could be stored as meter, inches etc. Moreover, the Range values are not always present in the data. Furthermore, we can not assume that if Linked Data resources do not store some attribute values, they are erroneous. Therefore relying on a single Range value for a property does not work in reality because the basic technique is too strict. On the other hand, in our proposal we divide Linked Data resources in groups and handle each group differently. It increases error data identification efficiency.

In Figure 6.1, the Phase 1 erroneous [S-O] pairs are shown in the shaded boxes.

6.3.2 Value Pattern Based Error Detection

The lower half of Figure 6.1 shows the work-flow of the value pattern based error detection. Here, we take groupwise [S-O] pairs that Phase 1 does not consider as candidates of erroneous [S-O] pairs. We considered such [S-O] pairs as mixed [S-O] pairs, because they still might hold some erroneous pairs. In this phase, we utilize the LV(O) to detect the error candidates. The 2nd column of Table 6.2 shows literal values for some Os. First, we use the process called Value Pattern Collector, which stores the Groupwise Value and the Groupwise Dependency. This information helps the next process, called P2 Error Detector, to detect candidates of erroneous [S-O] pairs.

6.3.2.1 Value Pattern Collector

Value Pattern Collector has two sub-processes: Value Collector and Dependency Collector. With the Value Collector, we store the LV(O) of the [S-O] pairs of a group and decide their Outlier. With the Dependency Collector, we devise Dependency patterns between the Properties of S of the [S-O] pairs of a group and decide which [S-O] pairs violate the usual pattern and then predict the potential error. For example, the Dependency pattern helps to identify the error that the "death date" should not be later than the "birth date".

We already discussed the LV(O) (see Section 6.3.1.1). Below we describe the extraction of Dependency patterns. The Dependency pattern is measured by the literal values that each two frequent Properties hold.

First, we discuss frequent Properties, then frequent Property related literal values (PrLV) and then the Dependency pattern calculation. A frequent Property is p ∈

PRT(S), which is at least common for the β % of S of the [S-O] pairs of G. Then, to find the Dependency pattern, we check Property related literal values for each frequent Property p_i,p_j. So, we calculate

$$\text{PrLV}(S,p_i,G) = \{\text{LV}(?o_i) \mid [S\text{-}?o_i] \in G, <S,p_i,?o_i>\}$$

$$\text{PrLV}(S,p_j,G) = \{\text{LV}(?o_j) \mid [S\text{-}?o_j] \in G, <S,p_j,?o_j>\}$$

Then, for PrLV(S,p_i,G) and PrLV(S,p_j,G), we check three different trends: ">", "=", and "<" as

$$\text{TRN}(S,p_i,p_j,G) = \begin{cases} > & (\text{if PrLV}(S,p_i,G) > \text{PrLV}(S,p_j,G)) \\ < & (\text{if PrLV}(S,p_i,G) < \text{PrLV}(S,p_j,G)) \\ = & (\text{otherwise }) \end{cases}$$

Then, for each trend (">" or "=" or "<"), we calculate the Groupwise trend

$$\begin{aligned} \text{GTRN}(G,p_i,p_j,\text{“>”}) = \; & \mid \{\text{TRN}(S,p_i,p_j,G) \mid \forall\, [S\text{-}O] \in G, \\ & \text{TRN}(S,p_i,p_j,G)=\text{“>”}\}|/|G| \end{aligned}$$

$$\begin{aligned} \text{GTRN}(G,p_i,p_j,\text{“<”}) = \; & \mid \{\text{TRN}(S,p_i,p_j,G) \mid \forall\, [S\text{-}O] \in G, \\ & \text{TRN}(S,p_i,p_j,G)=\text{“<”}\}|/|G| \end{aligned}$$

$$\begin{aligned} \text{GTRN}(G,p_i,p_j,\text{“=”}) = \; & \mid \{\text{TRN}(S,p_i,p_j,G) \mid \forall\, [S\text{-}O] \in G, \\ & \text{TRN}(S,p_i,p_j,G)=\text{“=”}\}|/|G| \end{aligned}$$

If any of them is larger than a threshold γ, we consider that the particular trend holds the Dependency pattern for p_i and p_j. Therefore, after establishing such a Dependency pattern, if any [S-O] pair violates the trend for p_i and p_j, we consider it as a candidate of erroneous [S-O] pair. We check the Dependency pattern for each two frequent Properties. Therefore, the Dependency pattern-based errors can be only found when two frequent Properties are present for an Linked Data resource. Currently, we devise the Dependency pattern for the frequent Property that generates the values NUMBER and DATE/TIME.

6.3.2.2 P2 Error Detector

We detect groupwise potential erroneous [S-O] pairs for the LV(O) (i.e., literal value of Object) and their Dependencies. We first describe how we detect erroneous the [S-O] pairs for the LV(O). Then we describe the same for their Dependency patterns.

For the literal values, we can have groups where the LVTs are either STRING, NUMBER, DATE/TIME, URI, or a data type. Below we describe finding the candidates of erroneous [S-O] pairs for each of them.

- LVT(O) is a STRING, so we consider the LV(O) could follow some patterns (such as patterns of Zip Codes), or could follow a syntactic similarity. When values follow some patterns, we check the common patterns among most of the values and violating [S-O] pairs are considered as erroneous. Currently, we adapt very basic common pattern checking, such as whether literal values hold some special characters after certain intervals, etc. When values follow syntactic similarity, we check the number of characters each LV(O) holds. We find the Outlier number based on the IQR (described in the P1 Error Detector). However, as an Outlier factor, we consider 0.25 instead of 1.5 because we assume that the number of characters for the LV(O) will not vary a lot. Outliers holding [S-O] pairs are considered as error candidates.
- LVT(O) is NUMBER, DATE/TIME, or a data type, so we find erroneous [S-O] pairs for their redundancy and their Outlier values.
  - If [S-O] pairs hold duplicate Os (i.e., Subject), we consider duplicate Os in the [S-O] pairs as erroneous. The rationale behind this is that we assume the same Property (e.g., ont:birthDate) would not hold different O values.
  - For the remaining [S-O] pairs, we follow Outlier based error founding. Again we use the IQR to calculate the Outlier. For example, in Table 6.2, the [S-O] pair [rsc:Tsuchida_Bakusen−217.7] is considered as an erroneous [S-O] pair.
- LVT is a URI, so currently we do not do anything for such [S-O] pairs; however, they also could hold errors.

Considering all the above methods for different LVTs, we try to find possible candidate of erroneous [S-O] pairs for values.

On the other hand, for Groupwise Dependency patterns, we check which [S-O] pairs violate the Dependency and consider them as erroneous.

6.4 Experiment

We performed experiments on DBpedia v3.8. DBpedia is a type-annotated Linked Data that covers various literal value type [S-O] pairs.

We did not find already defined Classes and Properties that can directly be used in the experiment. Therefore, we derived Classes and Properties from the erroneous DBpedia RDF triples <Subject, Property, Object> (or <S, P, O>) that Acosta et al. manually assessed by employing crowds in their study [1]. In our experiments, we consider the erroneous RDF triples of Acosta et al. study as baseline triples. By executing ALDErrD, our observation was whether ALDErrD could identify the RDF triples that Acosta et al. marked as incorrect.

We collected Property by the P of RDF triple <S, P, O> and Class by C = { c | <S, rdf:type, c> ∈ DBpedia RDF triples }[10]. As mentioned in the Section 6.3, the framework works better, when we input it with the most specific Class. We consider the most specific Class that does not hold rdfs:subClassOf closure and holds fewer RDF triples. For example, the RDF triple <res:Rodrigo_Salinas, prp:birthPlace, res:Puebla_F.C.> has four Classes, ont:Person, ont:Agent, ont:Athlete, and ont:Soccer Player. But we consider the most specific Class for res:Rodrigo_Salinas as ont:SoccerPlayer because ont:SoccerPlayer does not have rdfs:subClassOf closure attachment and has fewer RDF triples than the other three Classes.

We used the Stanford Parser[11] to parse literal values of S and O of the [S-O] pairs. As the threshold, we set $\alpha = 0.05$, $\beta = 80$, and $\gamma = 0.8$. The ALDErrD hardware specifications were as follows: Intel®Core™i7-4770K central processing unit (CPU) 3.50 GHz based system with 16 GB memory. We loaded DBpedia dataset in Virtuoso (version 06.01.3127) triple-store, which was maintained in a network server. The execution time was depended on number of Linked Data resources hold by the input Class and Property. In Phase 1, the NE (Named Entity) finding for the literal value of Object (for details, see Section 6.3.1.1) required large amount of time. For each Linked Data resource, on an average (calculated by executing 3 times) the NE finding required 6.4 seconds, and the rest of part of Phase 1 required 1.7 seconds. On the other hand, in Phase 2, the value pattern collection and the syntactic pattern checking required large amount of time. The value pattern collection was depended on other properties of Linked

[10] We discarded the "yago" ontology
[11] http://nlp.stanford.edu/software/corenlp.shtml

Data resources (see Section 6.3.2.1), therefore the execution time got increased when Linked Data resources held large amount of properties. For example, to identify error candidates for University and their address, it required almost one day for some 1800 Linked Data resources.

In experiments, we acknowledge that errors can be judgmental and purpose driven. Therefore, evaluating an error detection framework is not easy. Moreover, calculating recall values for errors over a large dataset might not be plausible.

6.4.1 Experiment 1

The purpose of this experiment is to investigate whether both phases (Phase 1 and Phase 2) of ALDErrD can detect candidates of erroneous [S-O] pairs and whether those candidates were correct.

We describe the experimental result for four different Classes and Properties that were present in the baseline RDF triples. We picked them by considering them to be i). the most specific Class, ii). representative for many types of literal values and iii.) will not generate [S-O] pairs more than 2000 so that we can manually evaluate their qualities. We report the erroneous [S-O] pair finding investigations for the Phase 1 errors and the Phase 2 errors.

When we executed ALDErrD for a most specific Class c and a Property P, it collected [S-O] pairs from {<S, P, O> | <S, rdf:type, c>}. We considered an [S-O] pair is erroneous if literal value of O is not correct, which were generated for either types of errors: Type-1 and Type-2 Errors.

While the baseline RDF triples provided one [S-O] pair for each such triple, we identified more number of erroneous [S-O] pairs for the same one RDF triple-driven Class and Property. This is because, the ALDErrD identified errors are for all instances of a Class for a particular Property. The [S-0] pairs that belong to baseline triples, we evaluated them directly. However, for the newly identified [S-O] pairs that were not identified by Acosta et al. employed crowds, we evaluated them by engaging three *linked data experts* who have knowledge about DBpedia, DBpedia ontology. The engagement of linked data experts is only for evaluation purpose and they remain outside of the proposed framework. We considered each [S-O] pair's evaluation on the basis of "majority voting".

For the derived Classes and Properties, Table 6.3 shows error candidate detection at Phases 1 and 2 with their precisions and recalls. It also shows error classifications into Type-1 and Type-2 Errors. In the table, the 1st column shows the input Class and Property, the 2nd column shows the number of [S-O] pairs are found for the corresponding Class and Property. The 3rd column shows the total number of errors existence among the [S-O] pairs. The total number of errors existence were measured by the linked data experts. These are the gold standard errors. The 4th and the 5th columns show the detected errors in number. The results are divided into Phase 1 and Phase 2. As mentioned in Section 6.3, ALDErrD only identifies Type-1 Errors. We show them by their number for each phase. We show correctly identified errors just below the detected numbers in square brackets (i.e., []). After detecting the Type-1 errors, we also investigated them for Type-2 errors. The investigation revealed some reason behind the Type-1 Errors. The investigation results are shown just below horizontal bars. The Types-2 errors were shown for Erroneous Type, Erroneous Domain, Erroneous Range and Erroneous Property for each phase. We investigated the Type-2 Errors and those were verified by the linked data experts. The 6th and the 7th columns show the error detection precision for Phase 1 and Phase 2. The 8th column shows the error detection recall.

The investigation showed that the erroneous [S-O] pairs were found more in Phase 1 than Phase 2 (although for last two cases, Phase 2 did not have any error candidates). Moreover, Phase 1 achieved higher precision than those in Phase 2. In the experiment, Phase 1 was more effective in identifying Type-2 Errors, while Phase 2 was more effective in identifying Type-1 Errors. However, as mentioned in Section 6.3 that both types of error may co-exist for same [S-O] pairs, we also found them in the experiment. As an example, for input ont:University and prp:address, Phase 2 identified good number of (i.e., 55) errors which belong to both Type-2 Errors and Type-1 Errors.

In Phase 2, we mainly try to find erroneous pairs by their values. We found that Phase 2 correctly identified erroneous data. As an example, for input ont:School and prp:campusSize, Phase 2 identified at least 4 erroneous Linked Data resources that hold string pattern anomalies for their Object values. However, the values were sometimes very much diverse e.g., campus sizes are written in various ways such as with number of students (in text), area in square kilometer (in text), etc.,

TABLE 6.3: Error candidate detection at Phase 1 and 2 and their detection classification, precision and recall

	# of [S-O] pairs	**Total # of Errors Exist**	**# of Detected Error Candidates**		**Precision**		**Recall**
			@ Phase 1	**@ Phase 2**	**@ Phase 1**	**@ Phase 2**	
			Type-1 Errors [Identified Correctly] / **Type-2 Errors**	**Type-1 Errors [Identified Correctly]** / **Type-2 Errors**			
ont:School prp:campusSize	334	67	Detected: 9 [Correct: 7] / Erroneous Type : 4, Erroneous Domain : 4, Erroneous Range : 7, Erroneous Property : 4	Detected: 41 [Correct: 11] / Erroneous Type : 7, Erroneous Domain : 7, Erroneous Range : 7, Erroneous Property : 7	0.889	0.268	0.269
ont:University prp:address	1731	620	Detected: 17 [Correct: 5] / Type Error : 1, Erroneous Domain : 1, Erroneous Range : 5, Erroneous Property: 1	Detected: 55 [Correct: 55] / Erroneous Type : 10, Erroneous Domain : 9, Erroneous Range : 55, Erroneous Property : 9	0.294	1.000	0.097
ont:Holiday prp:date	700	19	Detected: 19 [Correct: 19] / Erroneous Type : –, Erroneous Domain : –, Erroneous Range : 19, Property Error: –	Detected: 0 [Correct: –] / Erroneous Type : –, Erroneous Domain : –, Erroneous Range : –, Erroneous Property : –	1.000	–	1.000
ont:County ont:currency	1261	23	Detected: 27 [Correct: 23] / Erroneous Type : 4, Erroneous Domain : –, Erroneous Range : 23, Erroneous Property: –	Detected: 0 [Correct: –] / Erroneous Type : –, Erroneous Domain : –, Erroneous Range : –, Erroneous Property : –	0.851	–	1.000

therefore automatic identification of such errors require human judgments. However, for input ont:University and prp:address, Phase 2 achieved good precision value but they are only for tiny portion of the erroneous data (recall value is 0.097).

In all of the cases, the Object values were quite expressive by the Range values, therefore when Range values existed, it was easy to find errors. Moreover, we found that large number of Linked Data resources do not keep Linked Data attributes (Domain, Range, Type etc). But such attribute attached Linked Data resources would help maintaining better quality data. Over such attribute attached Linked Data, it would be easy to identify error candidates.

While Acosta et al. engaged crowds to identify each single error manually, ALDErrD detects errors in bulk automatically – which is an advantage over the Acosta et al. strategy. Moreover, the identification of errors for different types of literal value data can be considered as supportive argument that ALDErrD will be scalable over different datasets because datasets are mainly varied for their datatypes.

6.4.2 Experiment 2

The purpose of this experiment is to compare the error detection performance between ALDErrD and a state-of-the-art Linked Data error detection system [59]. The state-of-the-art system (i.e., we call as *rule based system*) devised 17 rules which can be adapted for different Properties and Classes. For example, one rule states that for a Property (say, ont:isbn), if the Object value does not match "∧[0-9]5$", the Property and Object values of the RDF triple are erroneous. However, adapting those rules needs explicit investigation of the Property and Object values and then selection of the appropriate rule and possible value matching constraints. This rule based system has been the subject of experiments for various types of literal value of DBpedia data, so we used it in the performance comparison.

In the rule based system, authors did not provide exact erroneous RDF triples that they identified. Therefore, to compare both systems with the same data, we used the RDF triples that hold Object value related errors in the Acosta et al. study. Acosta et al. provided 364 erroneous RDF triples that have wrong Object values in their RDF triples. We check whether both systems can capture the erroneous triples.

Table 6.4: Erroneous RDF triple finding performance comparison between ALDErrD and the ruled based system.

Type	**# of Erroneous Triples**	**ALDErrD**		**Rule Based**	
		# of Errors Found	**Error Found %**	**# of Errors Found**	**Error Found %**
DATE	151	100	66	95	63
STRING	134	39	29	10	7
NUMBER	76	32	42	24	32
URL	3	1	33	1	33
	364	172	47	130	36

We executed ALDErrD for the most specific Class and Property and checked for erroneous [S-O] pairs. If the [S-O] pairs hold the Subject and Object element of RDF triples that Acosta et al. predicted as erroneous, we considered the pairs as “Found”; otherwise we considered them as “Not Found”.

Table 6.4 shows the performance comparison between ALDErrD and the rule based system. According to the Gold standard Object values that Acosta et al. provided, we categorized Object values into four types: DATE, STRING, NUMBER, and URI (shown in the 1st column). The type calculation was done by the NE (named entity) of the parsed output of the Object value (for details, see Section 6.3.1.1). The 2nd column shows the number of erroneous RDF triples held by each type. The 3rd and 4th columns show the number of erroneous triples found and the errors found % by the proposed framework, respectively. The 5th and 6th columns show the same result for the rule based framework.

The bottom row shows total number of erroneous triples used in the test and their identification, by the systems.

Both systems worked comparatively well on the DATE type erroneous RDF triples. Most of the cases of date problems were due to their duplicate values. For other types of erroneous RDF triples, ALDErrD worked well. For example, for STRING type, ALDErrD identified three times more erroneous data than the rule based system. Phase 2 identified those errors. In overall comparison between the two systems, ALDErrD performs 10% better.

In ALDErrD,the nearest-neighbor based attribute value checking and the nearest-neighbor based literal value checking effectively identify erroneous [S-O] pairs.

Moreover, while the proposed framework automatically finds candidates of erroneous RDF triples, the rule based system always requires rule adaptation. In ALDErrD, the use of the parser for the Object value and the heuristic on devising the LVT (literal value type) minimizes the adaptation overhead.

6.5 Discussion

In Linked Data Data quality assessment, ALDErrD addresses the research challenge that we described in Chapter 1 Section 1.2.2. That is about

- "Wrong Entries" where we described that the existing Linked Datasets are not always clean. The unclean data contain various type of errors. However, the contemporary Linked Data error detection frameworks like [34, 61, 64, 79, 102, 111] are focused to particular error types and can not assess Linked Dataset for wide-rage error possibilities. Therefore, identifying of Linked Data errors for various type of errors is an open research issue. The manual quality assessment is not a feasible technique. As a result, we find that Linked Data quality assessment system lack of an automated framework that can assess errors, irrespective of their data types.

The below we discuss about how we address the issue.

- In our framework, we adapt a novel nearest-neighbor based outlier detection technique that predict possible erroneous entries automatically. The framework is not susceptible to a specific data type errors. Thereby we address the research issue.

  The framework ALDErrD takes a class (such as Person) and a property (such as Birth Date) as input, and detects whether the Linked Data resources hold erroneous entries that belong to the input class and property. In its methodology, other than some parameter setting, the framework does not require further supervision. Therefore, we consider it as an automatic system.

  On the other hand, ALDErrD detects Linked Data errors for erroneous object value ("Type-1 Errors"), and erroneous Linked Data attributes (e.g., type, domain, ontology etc) ("Type-2 Errors"). These two types of errors cover all possible error generation possibilities. Considering the Linked Data generation and publishing methods (described in Section 1.1 in Chapter 1), data are generated in two different methods − manual and automatic. In both methods, errors can be generated for object values i.e., Type-1 Errors. In manual case, erroneous entries could be generated because of human errors, while in automatic case, erroneous entries could be generated because of faulty data extraction procedure. On the other hand, when data are published with some ontology, errors can be generated for Type-2 Errors. Therefore, if a Linked Data quality assessment framework can detect these two types of errors, it can cover almost all type errors assessment possibilities. For such a framework, the performance would be how effectively it detects the errors. The result shown in 6.3 shows that ALDErrD can detect both Type-1 Errors and and Type-2 Errors. Since the proposed framework detects both Type-1 and Type-2 errors, we consider that it works irrespective of any error type. Not only that, the proposed framework also performs better than the state-of-the-art framework like [59]. The result shown in Table 6.4 supports our statement. In overall comparison between the two systems, ALDErrD performs 10% better.

Therefore, we consider that our proposed framework can address the research issue quite effectively.

6.6 Summary

The Linked Data is a large knowledge base. However, such data hold the possibility of erroneous data gathering. To use the Linked Data effectively, error detection is a requirement. On the other hand, a significant portion of these Linked Data keep the type information which populates type-annotated Linked Data. The type annotated RDF triples gives the opportunity to identify erroneous data.

In this Chapter, we propose a framework that can identify possible candidates of erroneous data over the type-annotated Linked Data. The proposed framework assesses errors for both for Linked Data attributes, and Linked Data values. We identify possible error candidates assuming that the same type of Linked Data resources share commonalities, which complies the usual outlier detection methods.

Therefore, the framework ALDErrD automatically detects possible error patterns and predicts possible error candidates. Our proposed framework is free from manual intervention, does not require domain-level expertise or the same kind of data sources, and is suitable for any type of data. We experimented with our proposed framework over DBpedia erroneous RDF triple benchmark data and found the framework effectively predicts erroneous triples. We also compared our system with a state-of-the-art system and found that our system works better.

ALDErrD framework can be used to assess a Linked Dataset, which will build an enhanced quality Dataset. Such an enhanced quality dataset can be used various purpose, like we can use it to retrieve more credible information. Thereby, the proposed Linked Data information access frameworks BoTLRet (described in Chapter 3 and TLDRet (described in Chapter 4) can retrieve credible information over enhanced quality datasets. Therefore, all of the proposed frameworks BoTLRet, TLDRet and ALDErrD will support in successful use of Linked Data and fulfilling the Linked Data initiatives.

Chapter 7

Discussion

In this Chapter we discuss about our contribution (Section 7.1). We show that how the proposed frameworks support in Linked Data success. Later, we describe how future works can use our contributions (Section 7.2). Section 7.3 summarizes the Chapter.

7.1 Discussion

Linked Data or Semantic Web data are inference-enable data, which hold capability of data sharing, exposing, and connecting among the applications [14]. Success of Linked Data depends on how effectively they can be used by the users and applications. Usually usage of any data relies upon how easily they can be accessed, and how good the data are i.e., knowing of data quality. In this thesis, we investigated these two primary issues of Link Data – information access and data quality assessment – to leverage Linked Data usage, and thereby support in Linked Data success. The keyword-based Linked Data information access systems help users to expose and access their information need easily, while the automatic Linked Data quality assessment system enhances Linked Data's data credibility. The contributions are also in-line with the "Big Picture" that we stated in Chapter 1 Figure 1.3.

With the keyword-based Linked Data information access systems i.e, BoTLRet [83, 84, 86] (the basic framework) and TLDRet [85] (the extension), data users do not need to think of Linked Data's complex data structure in the query. As users are familiar and comfortable with natural language-based or keyword-based query technique [77, 91], we consider that users of our frameworks will find them easy. In our proposal, for a natural language like query, we either manually construct the keyword-based query by knowing the Linked Dataset vocabularies. Or, machine learning-based system like LiCord [88], and entity linker (e.g., []) between keywords and vocabularies are used to devise the keyword-based query.

In information access, we also consider that our (novel) greedy template generation technique for keywords reduced complexity of data link finding. It also solved the problem of unstable template management that existed in contemporary information access frameworks [30, 31, 66, 107]. The proposed frameworks perform the same level of performance as an exhaustive systems performs, which conforms completeness competency of the greedy template management technique. Moreover, template merging technique that we proposed conforms information access requirement over Linked Data that guides that Linked Data information access needs to be done holistically.

Furthermore, extension of the basic framework TLDRet incorporates temporal features in query. To our knowledge, TLDRet is one of a first systems over Linked Data that retrieves Linked Data information for various type of temporal features.

It can handle explicit temporal feature such as definite DATE/TIME, event followed temporal feature such as "World War I" and relative temporal feature such as "last century". With TLDRet, we can leverage capability of inference-enable data of Linked Data i.e., accessing data with more semantics. Therefore, we consider that the proposed frameworks can be effectively used on Linked Data information access.

On the other hand, while Linked Data currently hold a vast amount of incorrect entries and the data generation methods always hold the incorrect data generation possibilities, the quality assessment framework ALDErrD [87] can automatically assess Linked Dataset, irrespective of the error types. The proposed framework can cover almost all types of error assessment possibilities. Not only that, the proposed framework also performs better than the state-of-the-art framework like [59]. In overall comparison between the two systems, ALDErrD performs 10% better. Therefore, we consider that the proposed framework can be effectively be used on Linked Data quality assessment. It can enhance data credibility.

Finally, we consider that the proposed frameworks will encourage Linked Data users to use the data for various purposes. It will, to some extent, leverage Linked Data success.

7.2 Future Work

We investigated two primary research issues over Linked Data, they are (1.) information access and (2.) quality assessment. We proposed frameworks successfully handle these two issues. In future, we want to see some researches that will focus both issues together. In such a case, user can be able to retrieve information with the assessed data quality.

Furthermore, we understand that Linked Data are network-structured knowledge, which are potential in link identification among various data, and construction of data links. To hold this definition, Linked Data should be accessed from one dataset to another dataset. Although our proposed information access frameworks are good at accessing information in a single dataset, they are not adapted for multiple datasets. Future researches can utilize our information access frameworks, particularly the template management part, to see they can interlink multiple datasets.

On the other hand, the Linked Data quality assessment framework can be adapted to propose correct data entries. Currently, it can only assess a Linked Dataset and can detect possible erroneous entries. However, this framework can be extended. The erroneous entries can be compared with other same-type correct entries and can be proposed a recommender system for correct entries.

7.3 Summary

In this thesis, we discuss on our three different frameworks BoTLRet, TLDRet and ALDErrD, how they address different challenges for Linked Data information access and Linked Data quality assessment.

Chapter 8

Conclusion

In this Chapter we conclude our thesis.

In this thesis, we proposed three different frameworks BoTLRet, TLDRet and ALDErrD. The first two frameworks facilitate in easy and efficient information access over Linked Data, and the last framework facilitates in quality assessment of Linked Data.

In information access, we solved three different challenges: information access by hiding complex data structure of Linked Data, information access by a stable and defined strategy, and information access by incorporating temporal semantics. Our basic proposal (BoTLRet) depends on some templates, where we advised that how the template should be construed and used for Linked Data information access. To do this, we analyzed data structure of Linked Data and proposed those templates. Since templates were constructed in conformity of data structure of Linked Data, they are able to retrieve required information. Furthermore, we showed that how templates should be automatically managed by the Linked Data statistics. It solved the problem of unstable template generation that exists in the contemporary systems. We extended the basic framework to TLDRet which successfully incorporated temporal semantics of a query. We implemented both information access proposals and experimented with standard question sets. Experimental results show that proposed frameworks can retrieve information over Linked Data. Moreover, they out-performed the contemporary systems.

In quality assessment, we solved issue of automatic Linked Data quality assessment, irrespective of any specific type of error. To detect the possible errors, we adapted an unsupervised nearest-neighbor based outlier detection technique. So, error detection required a similarity/distance measurement defined between/among the data. Since Linked Data are multivariate data, because data hold multiple attributes such type information, domain information, range information, etc., we adapted the error detection for "multivariate" data. We showed that the quality assessment framework covers all possible error generation possibilities. We implemented the framework. Experimental results showed that ALDErrD outperformed the state-of-the-art framework.

Linked Data vision heavily relies upon how effectively the data can be used by various applications. Usually usage of any data depends on how easily they can be accessed, and how good the data are. In this thesis, we contributed on these two and proposed frameworks which should leverage the Linked Data success.

Bibliography

[1] M. Acosta, A. Zaveri, E. Simperl, D. Kontokostas, S. Auer, and J. Lehmann. Crowdsourcing linked data quality assessment. In *Proceedings of the 12th International Semantic Web Conference*, pages 260–276, 2013.

[2] C. C. Aggarwal and H. Wang. Graph data management and mining: A survey of algorithms and applications. In C. C. Aggarwal and H. Wang, editors, *Managing and Mining Graph Data*, volume 40 of *Advances in Database Systems*, pages 13–68. Springer, 2010.

[3] C. C. Aggarwal and C. Zhai, editors. *Mining Text Data*. Springer, 2012.

[4] N. Aggarwal and P. Buitelaar. A system description of natural language query over dbpedia. In *Proceedings of Interacting with Linked Data (ILD 2012)*, pages 96–99, 2012.

[5] Z. Akar, T. G. Halaç, E. E. Ekinci, and O. Dikenelli. Querying the web of interlinked datasets using void descriptions. In *Proceedings of the 21st World Wide Web Conference Workshop on Linked Data on the Web*, pages 133–138, 2012.

[6] K. Alexander, M. Hausenblas, and J. Zhaox. Describing linked datasets - on the design and usage of voiD, the 'vocabulary of interlinked datasets'. In *Proceedings of the 18th International World Wide Web Conference Workshop on Linked Data on the Web*, 2009.

[7] S. Amit. Introducing the knowledge graph: things, not strings. https://googleblog.blogspot.jp/2012/05/introducing-knowledge-graph-things-not.html, 2012.

[8] G. Antoniou and F. v. Harmelen. *A Semantic Web Primer, 2Nd Edition (Cooperative Information Systems)*. The MIT Press, 2 edition, 2008.

[9] K. Bereta, P. Smeros, and M. Koubarakis. Representation and querying of valid time of triples in linked geospatial data. In *Proceedings of the 10th European Semantic Web Conference*, pages 259–274, 2013.

[10] T. Berners-Lee. *Linked Data - Design Issues*, 2006. http://www.w3.org/DesignIssues/LinkedData.html.

[11] T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific American*, 284(5):34–43, May 2001.

[12] G. Bhalotia, A. Hulgeri, C. Nakhe, S. Chakrabarti, and S. Sudarshan. Keyword searching and browsing in databases using BANKS. In *Proceedings of the 18th International Conference on Data Engineering*, pages 431–440, 2002.

[13] V. Bicer, T. Tran, A. Abecker, and R. Nedkov. KOIOS: Utilizing semantic search for easy-access and visualization of structured environmental data. In *Proceedings of the 10th International Semantic Web Conference*, pages 1–16, 2011.

[14] C. Bizer, T. Heath, and T. Berners-Lee. Linked data - the story so far. *Internationl Journal of Semantic Web Information System*, 5(3):1–22, 2009.

[15] S. Börzsönyi, D. Kossmann, and K. Stocker. The skyline operator. In *Proceedings of the 17th International Conference on Data Engineering, April 2-6, 2001, Heidelberg, Germany*, pages 421–430, 2001.

[16] M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander. Optics-of: Identifying local outliers. In *Proceedings of the Third European Conference on Principles of Data Mining and Knowledge Discovery*, PKDD '99, pages 262–270, 1999.

[17] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander. Lof: Identifying density-based local outliers. *SIGMOD Rec.*, 29(2):93–104, May 2000.

[18] N. Bruno, N. Koudas, and D. Srivastava. Holistic twig joins: optimal XML pattern matching. In *Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data, Madison, Wisconsin, June 3-6, 2002*, pages 310–321, 2002.

[19] V. Bryl and C. Bizer. Learning conflict resolution strategies for cross-language wikipedia data fusion. In *Proceedings of the Companion Publication*

of the 23rd International Conference on World Wide Web Companion, pages 1129–1134, 2014.

[20] E. Cabrio, A. P. Aprosio, J. Cojan, B. Magnini, F. Gandon, and A. Lavelli. Qakis qald-2. In *Proceedings of Interacting with Linked Data*, pages 87–95, 2012.

[21] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. *ACM Comput. Surv.*, 41(3):15:1–15:58, July 2009.

[22] A. X. Chang and C. Manning. SUTime: A library for recognizing and normalizing time expressions. In *Proceedings of the 8th International Conference on Language Resources and Evaluation*, pages 23–25, 2012.

[23] J. Chen, W. Hsu, M. L. Lee, and S.-K. Ng. Nemofinder: Dissecting genome-wide protein-protein interactions with meso-scale network motifs. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 106–115, 2006.

[24] L. Chen, A. Gupta, and M. E. Kurul. Stack-based algorithms for pattern matching on dags. In *Proceedings of the 31st International Conference on Very Large Data Bases*, pages 493–504, 2005.

[25] G. Cheng and Y. Qu. Searching linked objects with falcons: Approach, implementation and evaluation. *International Journal on Semantic Web Inforamation System*, 5(3):49–70, 2009.

[26] J. Cheng, J. X. Yu, B. Ding, P. S. Yu, and H. Wang. Fast graph pattern matching. In *Proceedings of the 24th International Conference on Data Engineering*, pages 913–922, 2008.

[27] L. Clark. SPARQL views: A visual SPARQL query builder for drupal. In *Proceedings of the ISWC 2010 Posters & Demonstrations Track: Collected Abstracts*, 2010.

[28] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, Sept. 1995.

[29] M. A. Covington. A dependency parser for variable-word-order languages. *Derohanes (eds.), Computer assisted modeling on the IBM*, 3090:799–845, 1992.

[30] D. Damljanovic, M. Agatonovic, and H. Cunningham. Natural language interfaces to ontologies: Combining syntactic analysis and ontology-based lookup through the user interaction. In *Proceedings of the 7th International Conference on The Semantic Web*, pages 106–120, 2010.

[31] D. Damljanovic, M. Agatonovic, and H. Cunningham. FREyA: An interactive way of querying linked data using natural language. In *Proceedings of the 1st Workshop on Question Answering over Linked Data (QALD-1), Collocated with the 8th Extended Semantic Web Conference*, pages 125–138, 2011.

[32] L. Derczynski and R. J. Gaizauskas. A corpus-based study of temporal signals. *Computing Research Repository*, abs/1203.5066, 2012.

[33] L. Ding, T. W. Finin, A. Joshi, R. Pan, R. S. Cost, Y. Peng, P. Reddivari, V. Doshi, and J. Sachs. Swoogle: A search and metadata engine for the semantic web. In *Proceedings of the 13th ACM Conference on Information and Knowledge Management*, pages 652–659, 2004.

[34] D. Fleischhacker, H. Paulheim, V. Bryl, J. Völker, and C. Bizer. Detecting errors in numerical linked data using cross-checked outlier detection. In *Proceedings of the 12th International Semantic Web Conference, ISWC 2014*, pages 357–372, 2014.

[35] A. Freitas, J. Oliveira, S. O'Riain, E. Curry, and J. Pereira da Silva. Treo: best-effort natural language queries over linked data. In *Proceedings of the 16th International Conference on Applications of Natural Language to Information Systems*, pages 286–289, 2011.

[36] E. B. Fry, J. E. Kress, and D. L. Fountoukidis. *The Reading Teacher's Book of Lists*. Englewood Cliffs, NJ: Prentice Hall, 1993.

[37] M. Gamon, A. Aue, S. Corston-Oliver, and E. Ringger. Pulse: Mining customer opinions from free text. In *Proceedings of the 6th International Conference on Advances in Intelligent Data Analysis*, pages 121–132, 2005.

[38] O. Ganea, M. Ganea, A. Lucchi, C. Eickhoff, and T. Hofmann. Probabilistic bag-of-hyperlinks model for entity linking. In *Proceedings of the 25th International Conference on World Wide Web*, pages 927–938, 2016.

[39] C. Gutierrez, C. Hurtado, and R. Vaisman. Temporal rdf. In *Proceedings of the 2nd European Semantic Web Conference*, pages 93–107, 2005.

[40] L. Han, T. Finin, and A. Joshi. Gorelations: An intuitive query system for DBpedia. In *Proceedings of the 1st Joint International Semantic Technology Conference*, pages 334–341, 2011.

[41] S. Harris and A. Seaborne. *SPARQL 1.1 Query Language*. W3C Recommendation, 2013. http://www.w3.org/TR/sparql11-query/.

[42] O. Hartig, C. Bizer, and J. C. Freytag. Executing sparql queries over the web of linked data. In *Proceedings of the 10th International Semantic Web Conference*, pages 293–309, 2009.

[43] D. Hawkins. *Identification of outliers*. Chapman and Hall, 1980.

[44] H. He, H. Wang, J. Yang, and P. S. Yu. BLINKS: ranked keyword searches on graphs. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, Beijing, China, June 12-14, 2007*, pages 305–316, 2007.

[45] S. He, S. Lui, Y. Chen, G. Zhou, K. Liu, and J. Zhao. Casia@qald-3:a question answering system over linked data. Work. Multilingual Question Answering over Linked Data (QALD-3), 2013. See Online Working Notes at www.clef2013.org.

[46] T. Heath and C. Bizer. *Linked Data: Evolving the Web into a Global Data Space*. Morgan & Claypool Publishers, 2011.

[47] B. Hyland, G. Atemezing, and B. Villazn-Terrazas. *Best Practices for Publishing Linked Data*, 2014. http://www.w3.org/TR/2014/NOTE-ld-bp-20140109/.

[48] A. Inokuchi, T. Washio, and H. Motoda. An apriori-based algorithm for mining frequent substructures from graph data. In *Principles of Data Mining and Knowledge Discovery, 4th European Conference, PKDD 2000*, pages 13–23, 2000.

[49] A. K. Jain and R. C. Dubes. *Algorithms for Clustering Data*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1988.

[50] H. Kanayama and T. Nasukawa. Unsupervised lexicon induction for clause-level detection of evaluations. *Natural Language Engineering*, 18(1):83–107, 2012.

[51] E. Kaufmann, A. Bernstein, and L. Fischer. NLP-Reduce: A nave but domain-independent natural language interface for querying ontologies. In *Proceedings of the 4th European Semantic Web Conference*, 2007.

[52] A. Khan, Y. Wu, and X. Yan. Emerging graph queries in linked data. In *IEEE 28th International Conference on Data Engineering (ICDE 2012), Washington, DC, USA (Arlington, Virginia), 1-5 April, 2012*, pages 1218–1221, 2012.

[53] A. Khan, X. Yan, and K. Wu. Towards proximity pattern mining in large graphs. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data*, pages 867–878, 2010.

[54] H. Khrouf, V. Milicic, and R. Troncy. EventMedia live: Exploring events connections in real-time to enhance content. In *Proceedings of the Semantic Web Challenge, at the 11th International Conference on The Semantic Web*, 2012.

[55] S. Kim, K. Toutanova, and H. Yu. Multilingual named entity recognition using parallel data and metadata from wikipedia. In *Proceedings of the 50th Annual Meeting on Association for Computational Linguistics*, pages 694–702, 2012.

[56] S. Kiritchenko, X. Zhu, and S. M. Mohammad. Sentiment analysis of short informal texts. *Journal of Artificial Intelligence Research (JAIR)*, 50:723–762, 2014.

[57] T. P. Klammer, M. R. Schulz, and A. D. Volpe. *Analyzing English Grammar*. Longman, Longman :, 2009.

[58] D. Kontokostas, M. Brümmer, S. Hellmann, J. Lehmann, and L. Ioannidis. NLP data cleansing based on linguistic ontology constraints. In *Proceedings of the 11th International Conference*, pages 224–239, 2014.

[59] D. Kontokostas, P. Westphal, S. Auer, S. Hellmann, J. Lehmann, R. Cornelissen, and A. Zaveri. Test-driven evaluation of linked data quality. In

Proceedings of the 23rd International Conference on World Wide Web, pages 747–758, 2014.

[60] M. Kuramochi and G. Karypis. Finding frequent patterns in a large sparse graph*. *Data Mining and Knowledge Discovery*, 11(3):243–271, 2005.

[61] J. Lehmann and L. Bühmann. ORE - A tool for repairing and enriching knowledge bases. In *Proceedings of the 9th International Semantic Web Conference*, pages 177–193, 2010.

[62] J. Lehmann and L. Bühmann. AutoSPARQL: Let users query your knowledge base. In *Proceedings of the 8th Extended Semantic Web Conference*, pages 63–79, 2011.

[63] J. Lehmann, T. Furche, G. Grasso, A.-C. N. Ngomo, C. Schallhart, A. Sellers, C. Unger, L. Bühmann, D. Gerber, K. Höffner, D. Liu, and S. Auer. Deqa: deep web extraction for question answering. In *Proceedings of the 11th international conference on The Semantic Web*, pages 131–147, 2012.

[64] J. Lehmann, D. Gerber, M. Morsey, and A.-C. Ngonga Ngomo. Defacto - deep fact validation. In *Proceedings of the 11th International Conference on The Semantic Web - Volume Part I*, pages 312–327, 2012.

[65] C. Li, J. Han, G. He, X. Jin, Y. Sun, Y. Yu, and T. Wu. Fast computation of simrank for static and dynamic information networks. In *Proceedings of the 13th International Conference on Extending Database Technology*, pages 465–476, 2010.

[66] V. Lopez, E. Motta, and V. S. Uren. Poweraqua: Fishing the semantic web. In *Proceedings of the 4th Extended Semantic Web Conference*, pages 393–410, 2006.

[67] V. Lopez, C. Unger, P. Cimiano, and E. Motta. Evaluating question answering over linked data. *J. Web Sem.*, 21:3–13, 2013.

[68] Y. Ma and J. Wu. Combining n-gram and dependency word pair for multi-document summarization. In *Proceedings of the 17th IEEE International Conference on Computational Science and Engineering*, pages 27–31, 2014.

[69] I. Mani and D. G. Wilson. Robust temporal processing of news. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, pages 69–76, 2000.

[70] C. D. Manning, P. Raghavan, and H. Schütze. *An Introduction to Information Retrieval.* Cambridge University Press, 2009.

[71] M. G. Matthias Wendt and H. D̈wiger. Linguistic modeling of linked open data for question answering. In *Proceedings of Interacting with Linked Data (ILD 2012)*, pages 75–86, 2012.

[72] E. Meij, M. Bron, L. Hollink, B. Huurnink, and M. de Rijke. Mapping queries to the linking open data cloud: A case study using dbpedia. *Journal of Web Semantics special issue on Semantic Search*, 9(4), 2011.

[73] P. N. Mendes, M. Jakob, A. García-Silva, and C. Bizer. Dbpedia spotlight: Shedding light on the web of documents. In *Proceedings of the 7th International Conference on Semantic Systems*, pages 1–8, 2011.

[74] P. N. Mendes, H. Mühleisen, and C. Bizer. Sieve: linked data quality assessment and fusion. In *Proceedings of the 2012 Joint EDBT/ICDT Workshops, Berlin, Germany, March 30, 2012*, pages 116–123, 2012.

[75] T. Nasukawa and T. Nagano. Text analysis and knowledge mining system. *IBM Systems Journal*, 40(4):967–984, 2001.

[76] T. Niesler and P. C. Woodland. Variable-length categoryn-gram language models. *Computer Speech and Language*, 13(1):99–124, 1999.

[77] X. Ning, H. Jin, W. Jia, and P. Yuan. Practical and effective IR-style keyword search over semantic web. *Journal of Information Processing and Management*, 45(2):263–271, 2009.

[78] E. Parzen. On estimation of a probability density function and mode. *The Annals of Mathematical Statistics*, 33(3):pp. 1065–1076, 1962.

[79] H. Paulheim and C. Bizer. Type inference on noisy RDF data. In *Proceedings of the 12th International Semantic Web Conference*, pages 510–525, 2013.

[80] D. L. Phuoc, A. Polleres, M. Hauswirth, G. Tummarello, and C. Morbidoni. Rapid prototyping of semantic mash-ups through semantic web pipes. In *Proceedings of the 18th International Conference on World Wide Web*, pages 581–590, 2009.

[81] E. Prud'hommeaux and A. Seaborne. SPARQL query language for RDF. W3C recommendation, W3C, Jan. 2008. http://www.w3.org/TR/2008/REC-rdf-sparql-query-20080115/.

[82] R. Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann Publishers, San Mateo, CA, 1993.

[83] M. Rahoman and R. Ichise. intesearch: An intelligent linked data information access framework. In *Proceedings of the 4th Joint International Conference*, pages 162–177, 2014.

[84] M.-M. Rahoman and R. Ichise. An automated template selection framework for keyword query over linked data. In *Proceedings of the 2nd Joint International Semantic Technology Conference*, pages 175–190, 2012.

[85] M.-M. Rahoman and R. Ichise. Tldret: A temporal semantic facilitated linked data retrieval framework. In *Proceedings of the 3rd Joint International Semantic Technology Conference*, pages 228–243, 2013.

[86] M.-M. Rahoman and R. Ichise. Automatic inclusion of semantics over keyword-based linked data retrieval. *IEICE Transactions on Information and Systems*, E97-D(11):2852–2862, 2014.

[87] M.-M. Rahoman and R. Ichise. Automatic erroneous data detection over type-annotated linked data. *IEICE Transactions on Data Engineering and Information Management*, E99-D(4):969–978, 2016.

[88] M.-M. Rahoman, T. Nasukawa, H. Kanayama, and R. Ichise. Licord: Language independent content word finder. In *Proceedings of the 11th International Conference on Hybrid Artificial Intelligence Systems*, pages 40–52, 2016.

[89] M. Ramadas, S. Ostermann, and B. Tjaden. Detecting anomalous network traffic with self-organizing maps. In *Proceedings of the 6th International Symposium on Recent Advances in Intrusion Detection*, pages 36–54, 2003.

[90] L. Ratinov, D. Roth, D. Downey, and M. Anderson. Local and global algorithms for disambiguation to wikipedia. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 1375–1384, 2011.

[91] M. Reichert, S. Linckels, C. Meinel, and T. Engel. Student's perception of a semantic search engine. In *Proceedings of the IADIS International Conference Cognition and Exploratory Learning in Digital Age*, pages 139–147, 2005.

[92] A. Rula, M. Palmonari, A. Harth, S. Stadtmüller, and A. Maurino. On the diversity and availability of temporal information in linked open data. In *Proceedings of the 11th international conference on The Semantic Web*, pages 492–507, 2012.

[93] E. Saquete, J. L. V. González, P. Martínez-Barco, R. Muñoz, and H. Llorens. Enhancing QA systems with complex temporal question processing capabilities. *Journal of Artificial Intelligence Research (JAIR)*, 35:775–811, 2009.

[94] F. Schilder and C. Habel. Temporal information extraction for temporal question answering. In *New Directions in Question Answering, Papers from the 2003 AAAI Spring Symposium TR SS-03-07*, pages 35–44, 2003.

[95] S. Shekarpour, S. Auer, A.-C. N. Ngomo, D. Gerber, S. Hellmann, and C. Stadler. Keyword-driven SPARQL query generation leveraging background knowledge. In *Proceedings of the 10th International Conference on Web Intelligence*, pages 203–210, 2011.

[96] K. Shinzato, T. Shibata, D. Kawahara, and S. Kurohashi. Tsubaki: An open search engine infrastructure for developing information access methodology. *Information and Media Technologies*, 7(1):354–365, 2012.

[97] R. Smith, A. Bivens, M. Embrechts, C. Palagiri, and B. Szymanski. Clustering approaches for anomaly-based intrusion detection. In *Proceedings of the Intelligent Engineering Systems through Artificial Neural Networks*, pages 579–584, 2002.

[98] T. Steiner, R. Troncy, and M. Hausenblas. How google is using linked data today and vision for tomorrow. In *Proceedings of Linked Data in the Future Internet at the Future Internet Assembly (FIA 2010)*, 2010.

[99] P.-N. Tan, M. Steinbach, and V. Kumar. *Introduction to Data Mining, (First Edition)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2005.

[100] M. Theobald, R. Schenkel, and G. Weikum. An efficient and versatile query engine for topx search. In *Proceedings of the 31st International Conference on Very Large Data Bases, Trondheim, Norway, August 30 - September 2, 2005*, pages 625–636, 2005.

[101] H. Tong, C. Faloutsos, B. Gallagher, and T. Eliassi-Rad. Fast best-effort pattern matching in large attributed graphs. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Jose, California, USA, August 12-15, 2007*, pages 737–746, 2007.

[102] G. Töpper, M. Knuth, and H. Sack. Dbpedia ontology enrichment for inconsistency detection. In *Proceedings of the 8th International Conference on Semantic Systems*, pages 33–40, 2012.

[103] T. Tran, P. Haase, and R. Studer. Semantic search - using graph-structured semantic models for supporting the search process. In *Proceedings of the 17th International Conference on Conceptual Structures: Conceptual Structures: Leveraging Semantic Technologies*, pages 48–65, 2009.

[104] T. Tran, H. Wang, and P. Haase. Hermes: Data web search on a pay-as-you-go integration infrastructure. *Journal of Web Semantics*, 7(3):189–203, 2009.

[105] R. Troncy, B. Malocha, and A. T. S. Fialho. Linking events with media. In *Proceedings of the 6th International Conference on Semantic Systems*, pages 1–4, 2010.

[106] G. Tummarello, R. Delbru, and E. Oren. Sindice.com: Weaving the open linked data. In *Proceedings of the 6th International Semantic Web Conference*, pages 552–565, 2007.

[107] C. Unger, L. Bühmann, J. Lehmann, A.-C. N. Ngomo, D. Gerber, and P. Cimiano. Template-based question answering over RDF data. In *Proceedings of the 21st World Wide Web Conference*, pages 639–648, 2012.

[108] G. J. G. Upton and I. Cook. *Understanding statistics.* Oxford University Press, Oxford :, 1996.

[109] P.-Y. Vandenbussche and C. Teissédre. Events retrieval using enhanced semantic web knowledge. In *Proceedings of the Workshop on Detection,*

Representation, and Exploitation of Events in the Semantic Web, pages 112–116, 2011.

[110] Y. Wang, M. Zhu, L. Qu, M. Spaniol, and G. Weikum. Timely yago: Harvesting, querying, and visualizing temporal knowledge from wikipedia. In *Proceedings of the 13th International Conference on Extending Database Technology*, pages 697–700, 2010.

[111] D. Wienand and H. Paulheim. Detecting incorrect numerical data in dbpedia. In *The Semantic Web: Trends and Challenges - 11th International Conference, ESWC 2014, Anissaras, Crete, Greece, May 25-29, 2014. Proceedings*, pages 504–518, 2014.

[112] E. Winkler. *Understanding Language: A Basic Course in Linguistics.* Continuum, Continuum :.

[113] M. Yahya, K. Berberich, S. Elbassuoni, and G. Weikum. Robust question answering over the web of linked data. In *Proceedings of the 22nd ACM International Conference on Conference on Information and Knowledge Management*, pages 1107–1116, 2013.

[114] X. Yan and J. Han. gspan: Graph-based substructure pattern mining. In *Proceedings of the 2002 IEEE International Conference on Data Mining*, pages 721–724, 2002.

[115] J. X. Yu, W. Qian, H. Lu, and A. Zhou. Finding centric local outliers in categorical/numerical spaces. *Knowledge Information System*, 9(3):309–338, Mar. 2006.

[116] A. Zaveri, D. Kontokostas, M. A. Sherif, L. Bühmann, M. Morsey, S. Auer, and J. Lehmann. User-driven quality evaluation of dbpedia. In *Proceedings of the 9th International Conference on Semantic Systems*, pages 97–104, 2013.

[117] G. Zenz, X. Zhou, E. Minack, W. Siberski, and W. Nejdl. From keywords to semantic queries-incremental query construction on the semantic web. *Journal of Web Semantics*, 7(3):166–176, Sept. 2009.

[118] L. Zhao and R. Ichise. Graph-based ontology analysis in the linked open data. In *Proceedings of the 8th International Conference on Semantic Systems*, pages 56–63, 2012.

[119] W. Zheng, L. Zou, and D. Zhao. Answering subgraph queries over large graphs. In *Proceedings of the 12th International Conference on Web-Age Information Management*, pages 390–402, 2011.

[120] L. Zou, L. Chen, M. T. Özsu, and D. Zhao. Answering pattern match queries in large graph databases via graph embedding. *VLDB J.*, 21(1):97–120, 2012.

Table 1: Question Answering over Linked Data 1 (QALD-1) DBpedia Questions

Q#	**Question**
1	Which companies are in the computer software?
2	Which telecommunications organizations are located in Belgium?
3	Give me the official websites of actors of the television show Charmed.
4	Give me the capitals of all U.S. states.
5	What are the official languages of the Philippines?
6	Who is the mayor of New York City?
7	Where did Abraham Lincoln die?
8	When was the Battle of Gettysburg?
9	Which countries have more than two official languages?
10	Is Michelle the wife of President Obama?
11	What is the area code of Berlin?
12	Which classics does the Millepede belong to?
13	In which country is the Limerick Lake?
14	Was Andrew Jackson involved in a war?
15	What is the profession of Frank Herbert?
16	Who is the owner of Universal Studios?
17	Which state of the United States of America has the highest dens?
18	Give me all cities in New Jersey with more than 100000 inhabitant.
19	What is the currency of the Czech Republic?
20	Which European countries are a constitutional monarchy?
21	How many monarchical countries are there in Europe?
22	Which European union members adopted the Euro?
23	Which presidents of the United States had more than three children?
24	What is the highest mountain in Germany?
25	Give me the homepage of Forbes.
26	Give me all soccer clubs in Spain.
27	What is the revenue of IBM?
28	Which states of Germany are governed by the Social Democratic?
29	In which films directed by Garry Marshall was Julia Roberts star?
30	Is proinsulin a protein?
31	Which museum exhibits The Scream?
32	Which television shows were created by Walt Disney?

Continued on next page

Table 1 – *Continued from previous page*

Q#	**Question**
33	Give me the creator of Goofy?
34	Through which countries flow the Yenisei river?
35	Is Egypts largest city also its capital?
36	Which monarchs of the United Kingdom were married to a German?
37	Who is the daughter of Bill Clinton married to?
38	Which states border Utah?
39	Which states of the united states possess native gold?
40	Who is the author of WikiLeaks?
41	Give me the designer of the Brooklyn Bridge.
42	Which bridges are of the same type as the Manhattan Bridge?
43	Which river does the Brooklyn Bridge cross?
44	Which locations have more than two caves?
45	Which mountain is the highest after the Annapurna?
46	What place is the highest place of Karakoram?
47	What did Bruce Carver die from?
48	When did Germany join the EU?
49	How tall is Claudia Schiffer?
50	In which country does the Nile start?

Table 2: Question Answering over Linked Data 1 (QALD-1) DBpedia Questions Corresponding Keywords

Q#	**Questions Corresponding Keywords**
1	{Company, Computer software, industry}
2	{Telecommunication, industry, Belgium}
3	{homepage, starring, Television Show, Charmed}
4	{capital, States of the United States}
5	{official language, Philippines}
6	{leader name, New York City}
7	{death place, Abraham Lincoln}
8	{date, Battle of Gettysburg}
9	{}
10	{spouse, Barack Obama}
11	{area code, Berlin}
12	{classis, Millipede}
13	{country, Limerick Lake}
14	{Presidents of the United States, Andrew Jackson, battle}
15	{profession, Frank Herbert}
16	{owner, Universal Studios}
17	{}
18	{}
19	{currency, Czech Republic}
20	{European countries, Constitutional monarchy}
21	{European countries, Constitutional monarchy}
22	{European Union member states, currency, Euro}
23	{}
24	{}
25	{homepage, Forbes}
26	{soccer club, Spain}
27	{IBM, revenue}
28	{States Of Germany, leader party, Social Democratic Party of Germany}
29	{Film, director, Garry Marshall, starring, Julia Roberts}
30	{Proinsulin, Protein}
31	{museum, The Scream}
32	{Television Show, creator, Walt Disney}

Continued on next page

Table 2 – *Continued from previous page*

Q#	**Questions Corresponding Keywords**
33	{creator, Goofy}
34	{country, Yenisei river}
35	{(Egypt, largest city) or (Egypt, capital)}
36	{Monarchs of the United Kingdom, spouse, birth place, Germany}
37	{Bill Clinton, child, spouse}
38	{}
39	{States of the United States, Mineral, Gold}
40	{author, Wikileaks}
41	{Brooklyn Bridge, designer}
42	{}
43	{Brooklyn Bridge, crosses}
44	{}
45	{}
46	{Karakoram, highest place}
47	{death cause, Bruce Carver}
48	{Germany, accessionEUdate}
49	{height, Claudia Schiffer}
50	{source country, Nile}

TABLE 3: Question Answering over Linked Data 2 (QALD-2) DBpedia Questions

Q#	Question
1	Which German cities have more than 250000 inhabitants?
2	Who was the successor of John F. Kennedy?
3	Who is the mayor of Berlin?
4	How many students does the Free University in Amsterdam have?
5	What is the second highest mountain on Earth?
6	Give me all professional skateboarders from Sweden.
7	When was Alberta admitted as province?
8	To which countries does the Himalayan mountain system extend?
9	Give me a list of all trumpet players that were bandleaders.
10	What is the total amount of men and women serving in the FDNY?
11	Who is the Formula 1 race driver with the most races?
12	Give me all world heritage sites designated within the past five years.
13	Who is the youngest player in the Premier League?
14	Give me all members of Prodigy.
15	What is the longest river?
16	Does the new Battlestar Galactica series have more episodes than the old one?
17	Give me all cars that are produced in Germany.
18	NOT GIVEN
19	Give me all people that were born in Vienna and died in Berlin.
20	How tall is Michael Jordan?
21	What is the capital of Canada?
22	Who is the governor of Texas?
23	Do Harry and William, Princes of Wales, have the same mother?
24	Who was the father of Queen Elizabeth II?
25	Which U.S. state has been admitted latest?
26	How many official languages are spoken on the Seychelles?
27	Sean Parnell is the governor of which U.S. State?
28	Give me all movies directed by Francis Ford Coppola.
29	Give me all actors starring in movies directed by and starring William Shatner.
30	What is the birth name of Angela Merkel?
31	Give me all current Methodist national leaders.
32	How often did Nicole Kidman marry?

Continued on next page

Table 3 – *Continued from previous page*

Q#	Question
33	Give me all Australian nonprofit organizations.
34	In which military conflicts did Lawrence of Arabia participate?
35	Who developed Skype?
36	What is the melting point of copper?
37	Give me all sister cities of Brno.
38	How many inhabitants does Maribor have?
39	Give me all companies in Munich.
40	List all boardgames by GMT.
41	Who founded Intel?
42	Who is the husband of Amanda Palmer?
43	Give me all breeds of the German Shepherd dog.
44	Which cities does the Weser flow through?
45	Which countries are connected by the Rhine?
46	Which professional surfers were born on the Philippines?
47	What is the average temperature on Hawaii?
48	In which UK city are the headquarters of the MI6?
49	Which other weapons did the designer of the Uzi develop?
50	Was the Cuban Missile Crisis earlier than the Bay of Pigs Invasion?
51	Give me all Frisian islands that belong to the Netherlands.
52	Who invented the zipper?
53	What is the ruling party in Lisbon?
54	What are the nicknames of San Francisco?
55	Which Greek goddesses dwelt on Mount Olympus?
56	When were the Hells Angels founded?
57	Give me the Apollo 14 astronauts.
58	What is the time zone of Salt Lake City?
59	Which U.S. states are in the same timezone as Utah?
60	Give me a list of all lakes in Denmark.
61	How many space missions have there been?
62	Did Socrates influence Aristotle?
63	Give me all Argentine films.
64	Give me all launch pads operated by NASA.
65	Which instruments did John Lennon play?

Continued on next page

Table 3 – *Continued from previous page*

Q#	**Question**
66	Which ships were called after Benjamin Franklin?
67	Who are the parents of the wife of Juan Carlos I?
68	How many employees does Google have?
69	Did Tesla win a Nobel prize in physics?
70	Is Michelle Obama the wife of Barack Obama?
71	When was the Statue of Liberty built?
72	In which U.S. state is Area 51 located?
73	How many children did Benjamin Franklin have?
74	When did Michael Jackson die?
75	Which daughters of British earls died in the same place they were born in?
76	List the children of Margaret Thatcher.
77	Who was called Scarface?
78	Was Margaret Thatcher a chemist?
79	Was Dutch Schultz a jew?
80	Give me all books by William Goldman with more than 300 pages.
81	Which books by Kerouac were published by Viking Press?
82	Give me a list of all American inventions.
83	How high is the Mount Everest?
84	Who created the comic Captain America?
85	How many people live in the capital of Australia?
86	What is the largest city in Australia?
87	Who composed the music for Harold and Maude?
88	Which films starring Clint Eastwood did he direct himself?
89	In which city was the former Dutch queen Juliana buried?
90	Where is the residence of the prime minister of Spain?
91	Which U.S. State has the abbreviation MN?
92	Show me all songs from Bruce Springsteen released between 1980 and 1990.
93	Which movies did Sam Raimi direct after Army of Darkness?
94	What is the founding year of the brewery that produces Pilsner Urquell?
95	Who wrote the lyrics for the Polish national anthem?
96	Give me all B-sides of the Ramones.
97	Who painted The Storm on the Sea of Galilee?
98	Which country does the creator of Miffy come from?

Continued on next page

Table 3 – *Continued from previous page*

Q#	**Question**
99	For which label did Elvis record his first album?
100	Who produces Orangina?

Table 4: Question Answering over Linked Data 2 (QALD-2) DBpedia Questions Corresponding Keywords

Q#	**Questions Corresponding Keywords**
1	{City/Town, country, Germany, populationTotal, 25000}
2	{successor, John F. Kennedy}
3	{leader name, Berlin}
4	{number of students, Vrije Universiteit}
5	{elevation, Mountain}
6	{(occupation, Skateboarding, birth place, Sweden) or (occupation, Skateboarding, birth place, country, Sweden)}
7	{Alberta, AdmittanceDate}
8	{country, Himalaya}
9	{instrument, Trumpet, occupation, Bandleader}
10	{strength, New York City Fire Department}
11	{Formula Once Racer, races}
12	{World Heritage Site, year}
13	{}
14	{band member, The Prodigy}
15	{length, River}
16	{}
17	{(Automobile, production, Germany) or (Automobile, assembly, Germany) or (Automobile, manufacturer, location country, Germany) or (Automobile, Automotive Companies of Germany)}
18	{}
19	{birth place, Vienna, death place, Berlin}
20	{height, Michael Jordan}
21	{capital, Canada}
22	{governor, Texas}
23	{}
24	{father, Queen Elizabeth II}
25	{state of the united states, admittancedate}
26	{official language, Seychelles}
27	{governor, Sean Parnell}

Continued on next page

Table 4 – *Continued from previous page*

Q#	Questions Corresponding Keywords
28	{Film, director, Francis Ford Coppola}
29	{Film, starring , William Shantner, director, William Shantner}
30	{birth name, Angela Merkel}
31	{religion, Methodism, Current National Leaders}
33	{(Nonprofit Organization, location country, Australia) or (Nonprofit Organization,location, country, Australia) }
34	{battle, T. E. Lawrence}
35	{founder, Skype}
36	{}
37	{}
38	{population total, Maribor}
39	{(company, location, Munich) or (company, headquarter, Munich) }
40	{publisher, GMT Games}
41	{founder, Intel}
42	{spouse, Amada Palmer}
43	{breed, German Shepherd}
44	{city, Weser}
45	{country, Rhine}
46	{occupation, Surfing, birth place , Philippines}
47	{}
48	{Secret Intelligence Service, headquarter, country, United Kingdom}
49	{Uzi , designer, designer, Weapon}
50	{}
51	{Frisian Islands, country, Netherlands}
52	{}
53	{leader party, Lisbon}
54	{nickname, San Francisco}
55	{Greek Goddesses, adobe, Mount Olympus}
56	{Hells Angels, founded}
57	{Apollo 14, mission}

Continued on next page

Table 4 – *Continued from previous page*

Q#	**Questions Corresponding Keywords**
58	{TimeZone, Slat Lake City}
59	{TimeZone, state of the united states, Utah}
60	{(Lake, country, Denmark)
	or (Lakes of Denmark) }
61	{space mission}
62	{}
63	{(Film, country, Argentina)
	or (ArgentineFilms)}
64	{Launch Pad, operator, NASA}
65	{instrument, Hohn Lennon}
66	{Ship namesake, Benjamin Franklin}
67	{Juan Carlos I, spouse, parent}
68	{number of employee, Google}
69	{Nikola Tesla, award}
70	{Barack Obama, spouse}
71	{Statue of Liberty, beginning date}
72	{}
73	{child, Benjamin Franlin}
74	{Michael Jackson, death date}
75	{}
76	{child, Margaret Thatcher}
77	{nickname, Scarface}
78	{profession, Margaret Thatcher}
79	{ethnicity, Dutch Schultz}
80	{Book, author, William Goldman, number of pages}
81	{Book, author, Jack Kerouac, publisher, Viking Press}
82	{American Inventions}
83	{elevation, Mount Everest}
84	{creator, Captain America}
85	{population total, capital, Australia}
86	{largest city, Australia}
87	{music composer, Harold and Maude}
88	{Film, starring , Clint Eastwood, director, Clint Eastwood}

Continued on next page

Table 4 – *Continued from previous page*

Q#	**Questions Corresponding Keywords**
89	{resting place, Juliana of the Netherlands}
90	{residence, prime minister of Spain}
91	{States Of The United States, postal abbreviation, MN}
92	{Song, artist, Bruce Springsteen, release date}
93	{}
94	{Pilsner Urquell, brewery, foundation}
95	{author, anthem, Poland}
96	{musical artist, Ramones, B-side}
97	{artist, The Storm on the Sea of Galilee}
98	{nationality, Miffy, creator}
99	{}
100	{(products, Orangina)}
	or (product, Orangina)

Table 5: Question Answering over Linked Data 1 (QALD-1) MusicBrainz Questions

Q#	Question
1	Which soundtrack did Clannad record?
2	Which bands were founded in 2010?
3	Are there any interviews with J.R.R. Tolkien?
4	How many bands are called Queen?
5	Which artists performed the song Over the Rainbow?
6	Give me all songs on the album Abbey Road.
7	What is the third song on the album In the Wee Small Hours?
8	How many bands are called Air?
9	Did Elvis Presley record albums that are jazz?
10	Which songs by Miles Davis are longer than 20 minutes?
11	Give me all members of The Muppets.
12	Give me all siblings of Michael Jackson.
13	Which albums contain interviews with Queen?
14	How many audiobooks by J.R.R. Tolkien are there?
15	Give me all soundtracks composed by John Williams.
16	Which person recorded the most singles?
17	When was Ludwig van Beethoven born?
18	Who wrote the lyrics for West Side Story?
19	Give me all Celldweller EPs.
20	What is the longest song by John Cage?
21	Which compilations does the song Waterloo by ABBA appear on?
22	When did Madonna get divorced with Guy Ritchie?
23	Give me the spouses of all members of The Beatles.
24	Are there bootleg CD recordings by Pink Floyd?
25	How many members does the largest group have?
26	Which rock album has the most tracks?
27	When was the daughter of Elvis Presley born?
28	How many EPs did Muse release?
29	Who did the vocals on the album Sabotage?
30	How many artists were born in 1966?
31	Is there a group called Michael Jackson?
32	Which bands was Robbie Williams a member of?

Continued on next page

Table 5 – *Continued from previous page*

Q#	Question
33	Was John Lennon married?
34	In which bands did Kurt Cobain play?
35	Which bands released more than 100 singles?
36	How many live albums did The Rolling Stones release?
37	Are there any live albums by the Beatles?
38	How many soundtracks did Hans Zimmer compose?
39	When were The Beatles founded?
40	Did Kurt Cobain have children?
41	Who is the creator of The Hobbit audiobook?
42	Give me the collaborations of Shakira.
43	Who created The Girl Is Mine (a CD single)?
44	Give me all audiobooks by Paul Auster.
45	How many jazz compilations are there?
46	How many children did John Lennon have?
47	Which member of The Beatles was married more than once?
48	List live albums by Nat King Cole.
49	Who was born on the same day as Frank Sinatra?
50	Since when does The Who exist?

Table 6: Question Answering over Linked Data 1 (QALD-1) MusicBrainz Questions Corresponding Keywords

Q#	**Questions Corresponding Keywords**
1	{release type, Type Sound track, Clannad, creator }
2	{artist Type, Type Group, begin Date, '2010-01-01' '2010-12-31'}
3	{release type, Type Interview, creator, J.R.R. Tolkien}
4	{artist Type, Type Group, Queen }
5	{Track, title, Over the Rainbow, creator}
6	{Album, release type, Type Album, Abbey Road, track List }
7	{}
8	{artist Type, Type Group, title, Air }
9	{Album, creator, Elvis Presley, jazz }
10	{}
11	{member of Band, to Artist, The Muppets}
12	{Michael Jackson, is Sibling, to Artist }
13	{release type, Type Interview, Queen, creator}
14	{release type, Type Audio book, creator, title, J.R.R. Tolkien }
15	{release type, Type Sound track, composer, title, John Williams}
16	{}
17	{Ludwig van Beethoven, begin Date}
18	{lyricist, West Side Story}
19	{creator, title, Celldweller , Abum, release type, Type EP}
20	{}
21	{track List, release type, Type Compilation, title, Waterloo, creator, title, ABBA }
22	{Madonna, married To, end Date, to Artist, Guy Ritchie }
23	{The Beatles, member of Band, to Artist, married To, to Artist}
24	{Album, release Status, Status Bootleg, creator, title, Pink Floyd }
25	{}
26	{}
27	{Artist, artist Type, Type Person, title, Elvis Presley, is Parent, to Artist, begin Date }
28	{release type, Type EP, title, Muse, creator }
29	{Album, title, Sabotage, vocal}
30	{Artist, artist Type, Type Person, begin Date, 1966}

Continued on next page

Table 6 – *Continued from previous page*

Q#	Questions Corresponding Keywords
31	{Artist, artist Type, Type Group, Michael Jackson }
32	{Robbie Williams, member of Band, to Artist}
33	{John Lennon, married To, to Artist }
34	{Kurt Cobain, member of Band, to Artist}
35	{}
36	{Album, release type, Type Live, creator, title, The Rolling Stones }
37	{Album, release type, Type Live, creator, title, The Beatles }
38	{Album, release type, Type Sound track, creator, title, Hans Zimmer }
39	{The Beatles, artist Type, Type Group, begin Date}
40	{is Parent, to Artist, Kurt Cobain}
41	{release type, Type Audio book, title, The Hobbit, creator}
42	{Shakira, collaborated With, to Artist}
43	{release type, Type Single, title, The Girl Is Mine, creator }
44	{Album, release type, Type Audio book, creator, title, Paul Auster }
45	{Album, release type, Type Compilation, jazz }
46	{John Lennon, is Parent, to Artist }
47	{}
48	{Album, release type, Type Live, creator, title, Nat King Cole }
49	{}
50	{The Who, begin Date }

Table 7: Question Answering over Linked Data 2 (QALD-2) MusicBrainz Questions

Q#	Question
1	Which groups were founded in 1924?
2	Who produced the album Dookie?
3	Which artists were born in July 1904?
4	Give me all artists who contributed to more than three collaborations.
5	How many tracks does La Isla Bonita have?
6	Who composed The Heroes of Telemark?
7	Which artists turn 50 on May 28, 2012?
8	Is Bugs a track on the album Vitalogy?
9	Who sang on the album Home Free?
10	Who made the lyrics of the song Mambo No. 5?
11	How long is Louder Than Words by Against All Authority?
12	Give me all albums which have the name of their artist as title.
13	Was the album Coming Home created by Lionel Richie?
14	How many singles did the Scorpions produce?
15	Who composed the song Coast To Coast?
16	Which bands broke up in 2000?
17	Which artist created Last Christmas?
18	Which artist made the albums Blackout and Circus?
19	With whom did Phil Collins work together?
20	Give me all bands whose name starts with Play.
21	How many singles did Peter Gabriel make?
22	To how many persons has Madonna been married?
23	Who was born on the same day as Tina Turner?
24	Which compilations does the song Last Christmas appear on?
25	How long is Last Christmas by Kate and Bob?
26	Give me all singles by Loona.
27	What kind of record is Hijo de la luna?
28	Who made the song (Everything I Do) I Do It for You?
29	Who made more than 30 albums?
30	Who was the husband of Whitney Houston?
31	Give me the death date of Ludwig van Beethoven.
32	How many pieces of work did Wolfgang Amadeus Mozart create?

Continued on next page

Table 7 – *Continued from previous page*

Q#	Question
33	When were the Dixie Chicks founded?
34	When was That Which Remains founded?
35	How many children did Bob Marley have?
36	When was That Which Remains founded?
37	Give me all artists who were both in a band and released a solo album.
38	In which albums does the song Because You Loved Me appear?
39	Is the song Peggy Sue on the Greatest Hits compilation by Buddy Holly?
40	What is Beyonce Knowles middle name?
41	Which albums of Elvis Presley have Elvis in their title?
42	How many live albums by Elvis Presley are there?
43	Which member of the Beatles has more than one children?
44	Give me all songs which are on live albums by Blondie.
45	Give me all artists born in March.
46	Give me all titles of singles by Phil Collins.
47	Which records did Robbie Williams record together with Martin Slattery?
48	Give me all soundtracks made by the Pet Shop Boys.
49	In which language does Nyusha sing?
50	On which singles did Robbie Williams collaborate with Nicole Kidman?
51	Did Kylie Minogue ever collaborate with Mariah Carey?
52	In which countries was the single Incomplete published?
53	Which group had 70 members?
54	What is the legal name of Loona?
55	Which member of Take That recorded the most albums?

Table 8: Question Answering over Linked Data 2 (QALD-2) MusicBrainz Questions Corresponding Keywords

Q#	**Questions Corresponding Keywords**
1	{Music Group, event, Birth, Date, 1924}
2	{release type, album, Dookie, producer}
3	{Solo Music Artist, Birth, Date, 1904 1907}
4	{Solo Music Artist, collaboration with}
5	{track, La Isla Bonita}
6	{sound track, The Heroes of Telemark, composer}
7	{MusicArtist, event, birth, date}
8	{album, Bugs, track}
9	{release type, album, singer, Home Free}
10	{lyricist, Mambor No. 5}
11	{Louder Than Words, maker, Against All Authority, duration}
12	{release type, album, maker, name}
13	{Coming Home, maker, name, Lionel Richie}
14	{release type, single, maker, name, Scorpions}
15	{track, Coast to Coast, maker}
16	{group , Death, Date, 2000}
17	{maker, Last Christmas}
18	{maker, album, Blackout}
19	{Collaborates With, Phil Collins}
20	{group , title}
21	{single, Peter Gabriel}
22	{Spouse Of, Madonna}
23	{birth, Date, birth, Date, Tina Turner}
24	{compilation, Last Christmas}
25	{duration, Last Christmas, Kate and Bob}
26	{single, Loona}
27	{release type, Hijo de la luna}
28	{maker, (Everything I Do) I Do It for You}
29	{album, maker}
30	{Spouse Of, Whitney Houston}
31	{death, Date, Ludwig van Beethoven}
32	{single, maker, Wolfgang Amadeus Mozart}

Continued on next page

Table 8 – *Continued from previous page*

Q#	**Questions Corresponding Keywords**
33	{Dixie Chicks, birth , Date}
34	{That Which Remains, birth , Date}
35	{Parent Of, Bob Marley}
36	{Morgenstein, birth , Date}
37	{release type , solo, member of, group}
38	{album, Because You Loved Me}
39	{ }
40	{ }
41	{Elvis, album, title}
42	{release type, live, Elvis Presely}
43	{parent of, member of , group, The Beatles}
44	{release type, live, Blondie}
45	{birth , Date, march}
46	{single, title, Phil Collins}
47	{record, Collaborates With, Robbie Williams, Martin Slattery}
48	{sound track, maker, Pet Shop Boys}
49	{ }
50	{Robbie Williams, Collaborates With, Nicole Kidman}
51	{Collaborates With, Kylie Minogue}
52	{ }
53	{ }
54	{ }
55	{single, member of, Take}