Human-centric Question Answering System over Multiple Different Knowledge Graphs

by

Nhuan Duc To

A thesis submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in

Software Engineering and Intelligent Systems

Department of Electrical and Computer Engineering

University of Alberta

© Nhuan Duc To, 2021

Abstract

The inception of Semantic Web including the Resource Description Framework (RDF) data model that provides a standard framework for publishing and sharing machine-understandable data on the Web enables the development of intelligent, semantically-oriented systems. Up to date, thousands of linked RDF datasets, also known as Knowledge Graphs (KGs), have been constructed and are available on the Web. Therefore, there is a need for Question-Answering (QA) systems that provide users with appropriate utilization of existing KGs and provide detailed and summarizing answers to the users' questions. Yet, the diversity of posing questions and the heterogeneity of KGs' schemas make the process of querying KGs quite challenging. Moreover, a single KG often does not provide sufficient information for a variety of questions.

In this work, we propose a methodology aiming at developing a QA system that can automatically construct KG queries, use information from multiple different KGs, combine obtained data, and handle conflicting information, summarize obtained data if suitable. To accomplish that, we introduce a set of methods for: aligning properties (determining degrees of equivalence) in different KGs; generating templates based on given question-SPARQL query pairs; and using generated templates for constructing specific SPARQL queries for answering newly asked questions. Besides usual/regular questions, the methods allow for asking questions that contain linguistic terms with imprecise meanings. They also allow for aggregating answers, generating linguistic summaries for some suitable questions, and handling conflicting in-

formation retrieved from multiple different KGs. Our running website at https://www.lingteqa.site/ illustrates such a system.

Preface

This thesis is based on the following publications:

Nhuan D. To, Marek Z. Reformat, and Ronald R. Yager, Linked Open Data: Uncertainty in Equivalence of Properties, EUSFLAT-2017, Warsaw, Poland, September 11-15, 2017.

Marek Z. Reformat, Ronald R. Yager, and Nhuan D. To, Defining Personalized Concepts for XBRL Using iPad Drawn Fuzzy Sets, Intelligent Systems in Accounting, Finance and Management, Vol. 25, Issue 2, 2018, pages 73-85.

Ronald R. Yager, Marek Z. Reformat, and Nhuan D. To, Drawing on the iPad to Input Fuzzy Sets with an Application to Linguistic Summaries, Information Sciences, Vol. 479, 2019, pages 277-291.

Nhuan D. To, Marek Z. Reformat, and Ronald R. Yager, OWA-based Summarization of Data using iPad-drawn Concepts, FUZZ-IEEE International Conference on Fuzzy Systems, New Orleans, USA, June 23-26, 2019.

Nhuan D. To and Marek Z. Reformat, 2020, October. Question-Answering System with Linguistic Terms over RDF Knowledge Graphs. In 2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC) (pp. 4236-4243). IEEE.

Nhuan D. To, Marek Z. Reformat, and Ronald R. Yager, 2021, July. Question-Answering System with Linguistic Summarization. In 2021 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE) (pp. 1-8). IEEE.

Nhuan D. To and Marek Z. Reformat, 2021, August. Data Fusion in Question Answering Systems over Multiple-Knowledge Bases. In 19th World Congress of the International Fuzzy Systems Association (IFSA), 12th Conference of the European Society for Fuzzy Logic and Technology (EUSFLAT), and 11th International Summer School on Aggregation Operators (AGOP)

(pp. 227-234). Atlantis Press.

To my family

For their unconditional love and support.

Acknowledgements

First of all, I would like to thank my advisory committee as a whole, Dr. Marek Reformat, Dr. Witold Pedrycz, and Dr. Petr Musilek. Thank you for your time and advice.

I cannot express how grateful I am to Dr. Marek Reformat for his tremendous support. Marek admitted me to the University of Alberta and relentlessly trained me to become a better researcher. He is a great person with extraordinary leadership, integrity, fairness, and management skills. I have learned more than enough from him. Thank you, Marek.

I am also incredibly thankful to professors and instructors who taught me, mentored me, or helped me at University of Alberta: Dr. Pedrycz, Dr. Reformat, Dr. Musilek, Dr. Dick, Dr. Barbosa, and Vince Davis. Thank you.

I would also like to thank the Vietnamese Government for granting me a four-year fellowship monitored by Vietnam International Education Development (VIED) to study in Canada from 2015 to 2019.

Finally, I am infinitely grateful to my family for encouraging me to pursue my Ph.D. study program at the University of Alberta.

Contents

List of Tables

List of Figures

Acronyms

ANN Artificial neural networks.

APIs Application Programming Interfaces.

CLS Context Satisfaction Level.

CNN Convolutional Neural Network.

CUWA Context-based User-defined Weighted Averaging.

DAG Directed Acyclic Graph.

GUI Graphical User Interface.

HTML Hypertext Markup Language.

HTTP Hypertext Transfer Protocol.

IR Information Retrieval.

IRI Internationalized Resource Identifier.

KGs Knowledge Graphs.

LOD Linked Open Data.

LSTM Long Short Term Memory.

NEs Named Entities.

NLP Natural Language Processing.

NP noun phrase.

OWA Ordered Weighted Aggregation.

OWL Web Ontology Language.

POS part-of-speech.

QALD Question Answering over Linked Data.

QAS Question-Answering system.

QASs Question-Answering systems.

RDF Resource Description Framework.

RDFS RDF Schemas.

RIM regular increasing monotone.

RNN Recurrent Neural Network.

SKOS Simple Knowledge Organization System.

SPARQL SPARQL Protocol and RDF Query Language.

SVM support vector machine.

tf-idf term frequency–inverse document frequency.

TiFS Tablet input of Fuzzy Sets.

URI Uniform Resource Identifier.

VP verb phrase.

W3C World Wide Web Consortium.

Chapter 1

Introduction

1.1 Motivation

One of the notable contributions of the Semantic Web is Resource Description Framework (RDF). It is a data model that provides a standard framework for representing data in a machine-understandable format and publishing it on the Web. Up to date, thousands of linked RDF datasets, also known as Knowledge Graphs (KGs), have been constructed and are available as a part of Linked Open Data (LOD)[1].

Full utilization of the data available on the Web requires methods and tools that allow users to use a natural means of communication, i.e., a natural language, to query LOD data repositories. Another aspect that makes the retrieval of information difficult is the distribution of relevant data among multiple repositories existing at different locations. The new methods should be capable of queries multiple data sources and aggregate obtained information.

So far, a standard way of retrieving RDF data from the Web is to execute queries represented in an RDF query language – SPARQL Protocol and RDF Query Language (SPARQL)[2]. Therefore, translating user's questions posed in a natural language into SPARQL queries is the first step in developing an RDF Question-Answering (QA) system. Additionally, the QAS should find the information requested by users in various data sources – Knowledge Graphs. Yet, different KGs often use different vocabularies for describing items; and

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

[2] https://www.w3.org/TR/sparql11-query/

they may provide conflicting information about the same fact. Therefore, data fusion mechanisms able to handle the problems are needed to enhance the results of the QAS.

The recent advances in Natural Language Processing (NLP) and Semantic Web technologies provide better utilization of the available data and information. The user's quest for easy data accessing and the advancements in technologies have brought some developments and improvements in the category of Question-Answering systems (QASs) that answer end-users questions posed in a natural language [20][23][45].

Although it looks like a step towards a more user-friendly way of retrieving information, some more work is still needed. We could highlight a need to develop processes and methods for answering questions that include imprecise concepts and require additional processing of obtained results. Users would appreciate working with systems that allow them to ask comprehensive questions containing imprecise terms requiring subjective interpretation and provide them with the results in a condensed form. Therefore, designing and developing algorithms and methods, as parts of a QA system, that can accept questions asked in the user's natural language with user-defined imprecise terms and communicate results in the same form are essential steps towards constructing human-centric systems.

1.2 Objectives

A simplified architecture depicted in Figure 3.2(a) is of a currently existing QA system. Once a user asks a question, the QA system queries a data resource– could be a structured database as in the case of WDAqua-core0 [22], or an unstructured set of documents as in the case of the Wikipedia DrQA [13], and then it provides an answer to the user.

Our ultimate objective is to develop a set of methodologies and approaches that are necessary for constructing a system that is more human-friendly and human-oriented – a system that allows for using imprecise concepts, asking questions that require additional processing, and providing more summary-

like answers.

In other words, our objective is to develop a base for constructing a Human-centric QA system that would interact with end-users in English and cooperate with them in processing answers to their questions. In particular, we aim at building a QA system that can: 1) analyze asked questions to learn how to answer 'unseen' questions that are syntactically similar to the ones it has already 'seen'; 2) accept full-fledged English questions, rephrase them if necessary, then represent them in a form that facilitates retrieving relevant query templates from the repository of 'seen' questions; 3) populate retrieved query templates to construct queries executable over multiple different KGs; 4) cooperate with a user in defining linguistic terms (if any) so that questions contain such terms can be answered accordingly to the user's understanding of the terms; and 5) synthesize answers from information collected from multiple KGs and generate linguistic data summaries (if suitable) based on them.

By comparison with a 'traditional' QA system, our proposed system contains several extensions. These extensions are illustrated in Figure 3.2(b)

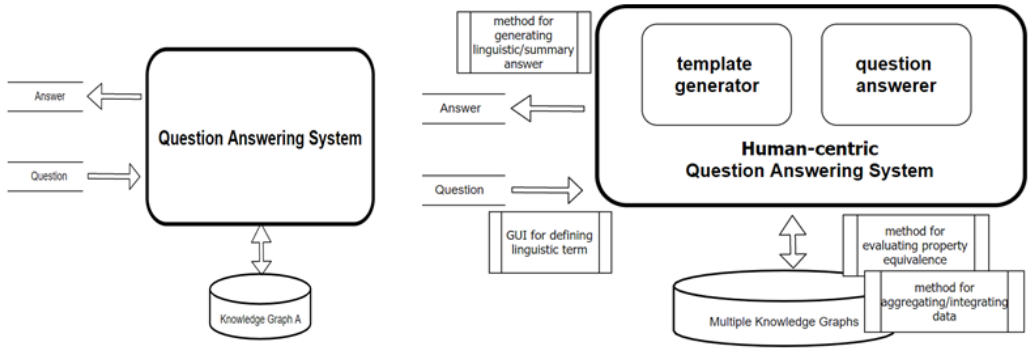


Figure 1.1: 'traditional' QA system (a); and Human-centric one (b).

1.3 Contributions

In this work, we aim at constructing methods and algorithms that allow us to address the previously stated objectives. We anticipate developing a methodology that comprises of a set of tools for: aligning properties, i.e., determining

degrees of equivalence between relations defined by different vocabularies used in various KGs; generating templates based on provided question-SPARQL query pairs; using generated templates for constructing specific queries executed over different KGs. The methods will allow for asking both ordinary questions and questions that contain linguistic terms with imprecise meanings. To address this, we develop an iPad-based application for entering fuzzy sets membership functions representing perceptions of users of linguistic terms and quantifiers.

The ability of the developed QA system to work with a large set of numerical data, also obtained from multiple sources, means we equip our system with answer-summarization techniques for aggregating query results and generating answers in the form of simple linguistic summaries.

In summary, the contributions of our work can be listed as a sequence of the following points.

1) A new method for generating question-query templates based on pairs of question-SPARQL query. The generated templates will then be used for translating newly asked questions into SPARQL queries.

2) A method for constructing SPARQL queries based on chosen KGs and asked questions using generated templates.

3) An algorithm for determining equivalence degrees between KGs' property. The obtained equivalent property pairs will then be used for collecting and fusing data from multiple different KGs.

4) A user-driven method for entering fuzzy sets defining linguistic terms and linguistic quantifiers applied for answering questions and generating linguistic summaries.

5) A novel methodology for fusing collected data from multiple different KGs. In particular, it includes

   – an introduction of the measure called *veracity* that is used for selecting a reliable data from a set of candidates, potentially conflicting

ones, retrieved from different data sources;

- two approaches for initializing trustworthiness of data sources: an expert-based approach inspired by Saaty's priority method [88], and a data-driven approach based on the degree of equivalence of RDF properties;
- an algorithm for updating the trustworthiness of data sources based on the results of a fusion process.

6) A human-centric methodology for summarizing collected data in form of an aggregated value or linguistic summary. More specifically, we focus on

   - introducing a context into basic operations on data, such as averaging or finding minimum and maximum values;
   - converting results of queries into simple linguistic answers that includes quantifiers and summarizers in a form of linguistic terms.

1.4 Thesis Outline

A Question-Answering System over multiple KGs is composed of multiple modules for: processing users' questions, generating query templates, constructing SPARQL queries over multiple RDF datastores and using different vocabularies, collecting data (answers to queries), processing collected data including fusion of data from multiple sources, and forming answers. Developing such a system requires techniques from both Semantic Web and Natural Language Processing. Furthermore, the application of fuzzy sets in developing QA systems allows for more user-friendly interfaces. This thesis introduces methodologies for developing such a user-friendly QA system. The following offers a concise summary of topics covered in each chapter and emphasizes the essential aspects of each of them.

Chapter 1 – *Introduction* – brings our motivations as well as objectives for developing a user-friendly QA system. It also states our anticipated contributions to the RDF Question-Answering Systems.

Chapter 2 – *Background Material* – provides brief introductions to such topics as Semantic Web, Natural Language Processing, and Fuzzy Set theory.

Chapter 3 –*Question Answering System: A Basic Version* – presents a basic of our *Analogical Problem Solving* based QA system, i.e., an approach, implementation and results; an overview state-of-the-art approaches towards Question-Answering systems. It also provides an overview on a proposed Human-Centric QA system with some advanced functionalities (let's call them extensions) such as a Graphical User Interface (GUI) for entering membership functions facilitating answering questions containing linguistic terms, fusing data collected from multiple KGs, generating linguistic summaries.

Stating from Chapter 4, we discuss each of the extensions in more detail. In particular,

Chapter 4 –*Ipad-based application for entering fuzzy sets defining linguistic terms* – describes an iPad-based software that is an easy procedure of defining linguistic terms – such as LARGE, MEDIUM, SMALL – and linguistic qualifiers – ALL, MOSTLY – that are suitable for answering questions with linguistic terms and generating linguistic summarizations.

Chapter 5 –*RDF Data Fusion* – provides methodologies for collecting and choosing true value from collected ones in multiple different Knowledge Graphs.

Chapter 6–*Data Summarization* – proposes novel methods for aggregating numeric and generating linguistic summaries.

Chapter 2

Background Material

2.1 Semantic Web technologies

The term "Semantic Web" originally coined by World Wide Web Consortium (W3C) and was formally introduced to the world by Sir Tim Berners-Lee [8]. It refers to the vision of the Web of linked data. Semantic Web technologies [1] enable people to create data stores on the Web, build vocabularies, and write rules for handling data.

2.1.1 Resource Description Framework

The W3C has introduced a graph-based data representation form called Resource Description Framework (RDF). It is a de-facto standard for representing semantically rich information on the Web. In RDF, a piece of information is represented as an RDF triple: subject-predicate-object. Asserting an RDF triple means that a relationship represented by a predicate holds between a subject and an object of that triple. The predicate itself is identified by a Uniform Resource Identifier (URI) and is also called a property. It is a special resource that defines a binary relation.

An RDF graph is a set of *triples*. A single RDF triple can be visualized as a *node-arc-node* link. A number of such triples that share subject nodes and object nodes constitute a Knowledge Graph. A simple RDF graph is depicted in Figure 2.1.

[1] https://www.w3.org/standards/semanticweb/

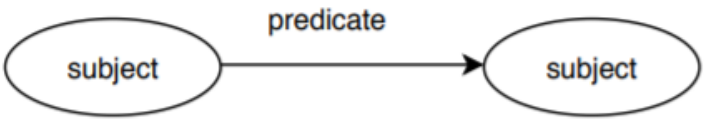


Figure 2.1: An RDF graph with one triple

There are three kinds of nodes in an RDF graph: *Internationalized Resource Identifier (IRI)*, *literals*, and *blank nodes*. IRIs and literals denoting items existing in the world, i.e., in a universe of discourse. They are called *resources*. Resources denoted by IRIs are named referent, and resources denoted by literals have datatype values such as strings, numbers, and dates. Literals that are language-tagged strings denote plain text expressed in a natural language.

2.1.2 Web Ontology Language

An ontology formally defines a common set of terms for describing and representing a domain (an area of knowledge). These terms are individuals (instances of objects), classes, properties (attributes and relations), restrictions, rules, and axioms. As a result, ontologies can introduce a sharable and reusable knowledge representation and can add new knowledge about the domain.

W3C offers a large palette of techniques to describe and define different forms of ontologies in a standard format. These include RDF Schemas (RDFS), Simple Knowledge Organization System (SKOS), Web Ontology Language (OWL)[2].

RDFS is a collection of terms we can use to define classes and properties for a specific application domain. In particular, to define classes we can use terms such as rdfs:Resource, rdfs:Class, rdfs:Literal, rdfs:Datatype, rdfs:subClassOf. To define properties, we can use terms such as rdfs:range, rdfs:domain, rdfs:subPropertyOf, rdfs:label and rdfs:comment. Some other utility terms are rdfs:seeAlso and rdfs:isDefinedBy.

SKOS provides a standard way to represent knowledge organization systems such as taxonomies and thesauri using the RDF. Encoding this informa-

[2]https://www.w3.org/standards/semanticweb/ontology

tion in RDF allows it to be passed between computer applications in an interoperable way. Concept is a fundamental element in any given knowledge organization systems (KOS). SKOS introduces the class skos:Concept, so that we can use it to state that a given resource is a concept. Labels on a given concept can be assigned by using label properties such as skos:prefLabel, skos:altLabel and skos:hiddenLabel. SKOS also provides some human-readable documentation properties defined for a given concept like skos:scopeNote, skos:definition, skos:example. To represent hierarchical structure of the KOS, we can use skos:broader and skos:narrower which can either be an is-a relationship or a part-of relationship. SKOS's specification can be found at W3C Recommendation 18 August 2009[3].

OWL[4] is a Semantic Web language designed to represent rich and complex knowledge about things, groups of things, and relations between things. OWL is a computational logic-based language such that knowledge expressed in OWL can be reasoned with by computer programs either to verify the consistency of that knowledge or to make implicit knowledge explicit. The purpose of OWL is exactly the same as RDF Schema; however, compared to RDFS, OWL provides us with the capability to express much more complex and richer relationships. Therefore, we can construct applications with a much stronger reasoning ability.

In OWL, owl:Thing is the root of all classes, it is also the base class of rdfs:Resource. To define a class, we can use owl:Class or rdfs:Class. It also gives us the ability to construct classes by using set operators (owl:intersectionOf, owl:unionOf, owl:complementOf), enumerating its instances(owl:oneOf), specifying a class is equivalent to another class (owl:equivalentClass), and specifying a class is disjoint from another class (owl:disjointWith). Furthermore, OWL provides owl:Restriction used together with owl:onProperty and other properties such as owl:allValuesFrom, owl:someValuesFrom, owl:hasValue to describe an anonymous class, which is defined by adding some restriction on some property.

[3]https://www.w3.org/TR/skos-reference/
[4]https://www.w3.org/OWL/

2.1.3 SPARQL: An RDF Query Language

SPARQL[5] is a query language that we can use to query the RDF data content, and SPARQL also provides a protocol that we need to follow if we want to query a remote RDF dataset. These are what SPARQL stands for: *SPARQL Protocol and RDF Query Language*. SPARQL features graph patterns, filters, unions, differences, optionals, aggregations, expressions, subqueries, ordering, etc.

A SPARQL query is executed against an RDF dataset that represents a collection of graphs. An RDF dataset contains one default graph, which does not have a name, and zero or more named graphs, where each named graph is identified by an IRI. By simply submitting a query, we should be able to directly get the answer for our information need.

A SPARQL endpoint is an interface that users can access to query a RDF dataset by using SPARQL query language. This endpoint could be a standalone or Web-based application that a user can work on. For example, *https://dbpedia.org/sparql* and *https://query.wikidata.org/sparql* are SPARQL endpoints of *DBpedia* and *Wikidata*, respectively. For applications, an endpoint takes the form of a set of Application Programming Interfaces (APIs) that can be used by the calling agent.

In general, SPARQL 1.1 provides four main forms of query:

- SELECT query
- ASK query
- DESCRIBE query
- CONSTRUCT query

Among these forms, the SELECT query and ASK query are the most frequently used in question answering systems. In addition, all these query forms are based on two basic SPARQL concepts: the triple pattern and the graph pattern.

[5]https://www.w3.org/TR/sparql11-query/

Triple Pattern: SPARQL is built the concept of triple pattern, which is written as subject, predicate and object, and has to be terminated with a full stop. Any or all of the subject, predicate and object values in a triple pattern can be a variable. For example, $< https://dbpedia.org/resource/Canada >$ $< https://dbpedia.org/ontology/capital >$ *?capital.* is a SPARQL's triple pattern whose subject and predicate are DBpedia's resources and its object is a variable, *?capital*. A variable in a triple pattern, which can be prefixed with either a ? character or a $ character, can be viewed as a placeholder that can match any value.

Graph Pattern: a collection of triple patterns that are within { and } called a graph pattern. For instance,

{*dbr*:*Canada* *dbo*:*capital* *?capital. ?capital* *rdfs*:*label* *?capitalLabel.* } is a graph pattern. Here dbr, dbo, and rdfs are prefixes defined as follows
PREFIX dbr: <https://dbpedia.org/resource/>
PREFIX dbo: <https://dbpedia.org/ontology/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT query

A generic structure of a SPARQL SELECT query is
base directive $BASE < URI >$ # list of prefixes
PREFIX pref: $< URI >$
...
result description
SELECT ...
graph to search
FROM ...
# query pattern
WHERE {
...
}
# query modifiers

ORDER BY ...

The *BASE* directive and a list of *PREFIX* are optional, and they are used for URI abbreviations. The *SELECT* clause specifies which variable bindings, or data items, should be returned from the query. As a result, it selects what information to return from the query result. The *FROM* clause, which is also optional, tells the SPARQL endpoint against which graph the search should be conducted. The *WHERE* clause contains graph patterns that specify the desired results; it tells the SPARQL endpoint what to query for in the underlying data graph. Value constraints specified by *FILTER* keyword are logical expressions that evaluate to boolean values when applied on values of bound variables can be added to the graph patterns. The optional query modifiers such as *ORDER BY...* and *LIMIT...* tell the SPARQL endpoint how to organize the query results.

CONSTRUCT query

Unlike *SELECT* query, *CONSTRUCT* query construct returns a new RDF graph containing query solutions. Its syntax is similar to the *SELECT* query's syntax. The only difference is the use of the keyword *CONSTRUCT* at the position of the keyword *SELECT*.

DESCRIBE query

Sometimes when querying an RDF we just don't know much about its vocabularies. If this is the case, we can ask a SPARQL query processor to describe the resource we want to know, and it is up to the processor to provide some useful information about the resource we have asked about. The *DESCRIBE* is the query used for that purpose. After receiving the query, a SPARQL processor creates and returns an RDF graph; the content of the graph is decided by the query processor, not the query itself. For example, the following *DESCRIBE* query:

DESCRIBE ?x
WHERE{?x rdfs:label 'Justin Trudeau'@en.}

when sent to *DBpedia* SPARQL endpoint will return information about subject whose English label is 'Justin Trudeau' in *DBpedia*; part of returning results is

Subject Item
dbr:2013_Liberal_Party_of_Canada_leadership_election
dct:subject
dbc:Justin_Trudeau
dbo:wikiPageWikiLink
dbr:Justin_Trudeau dbc:Justin_Trudeau
dbp:afterElection
dbr:Justin_Trudeau
dbp:winner
dbr:Justin_Trudeau

ASK query

SPARQL's *ASK* query is identified by the *ASK* keyword, and the query processor returns a *True* or *False* value, depending on whether the given graph pattern has any matches in the RDF graph or not. For instance, following *ASK* query:

ASK

WHERE{*dbr:Canada dbo:capital dbr:Ottawa.*}

when sent to *DBpedia* SPARQL endpoint will return a value of *True*.

2.1.4 Linked Data and Open Linked Data Cloud

Linked Data can be defined as a collection of interrelated datasets on the Web. Linked Data is used for large scale integration of, and reasoning on, data on the Web. Tim Berners-Lee proposed four principles in his 2006 Web architecture note as follows[6]: 1) Use URIs as names for things. 2) Use Hypertext Transfer Protocol (HTTP) URIs so that people can look up those names. 3) When someone looks up a URI, useful information should be provided using the standards (RDF*, SPARQL). 4) Include links to other URIs, so that users can discover more things.

[6]https://www.w3.org/DesignIssues/LinkedData.html

Also according to Sir Tim Berners-Lee, Linked Open Data (LOD)is Linked Data which is released under an open license, which does not impede its reuse for free. The goal of the Linking Open Data project is to build a data commons by making various open data sources available on the Web as RDF and by setting RDF links between data items from different data sources. As query results are structured data and not just links to Hypertext Markup Language (HTML) pages, they can be used within other applications. As of May 2020, there are 1301 datasets with 16283 links published in Linked Open Data cloud[7].

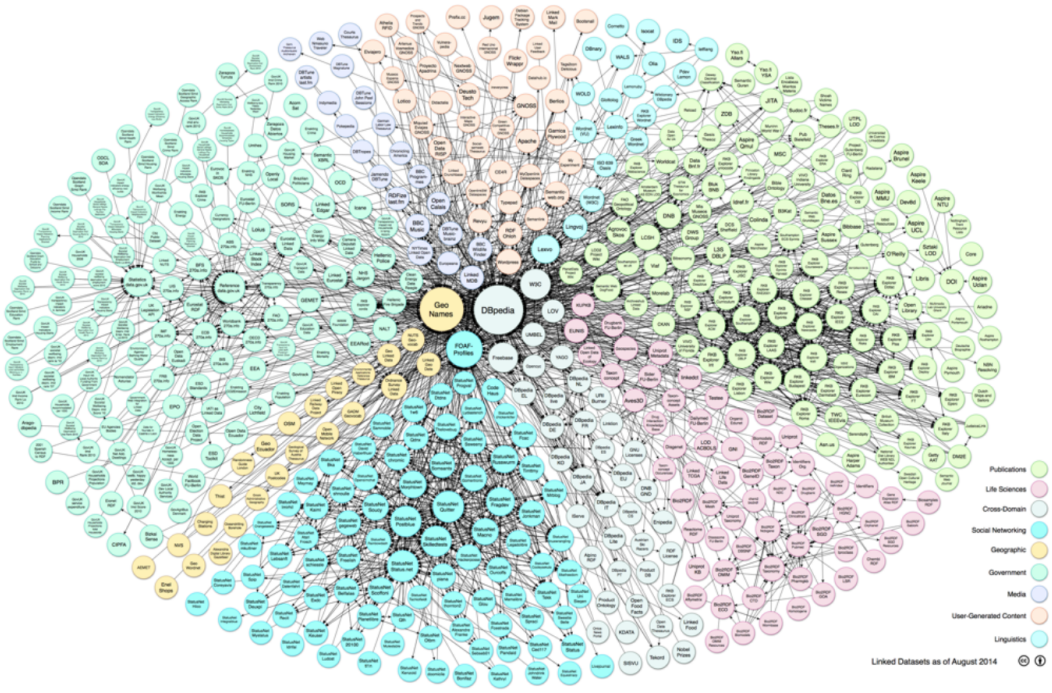


Figure 2.2: The Linked Open Data Cloud from lod-cloud.net

Many datasets in LOD cloud are enormous cross-domain datasets (also known as Knowledge Graphs-KGs), such as DBpedia[8] and Wikidata[9], containing million RDF triples.

DBpedia makes the content of Wikipedia available in RDF. It not only includes Wikipedia data, but also incorporates links to other datasets on the

[7]https://lod-cloud.net/
[8]https://wiki.dbpedia.org/about
[9]https://www.wikidata.org/wiki/Wikidata:Main_Page

Web, e.g., to Geonames. By providing those extra links applications may exploit the extra knowledge from other datasets when developing an application.

2.2 Natural Language Processing Tools and Techniques

In this section, we introduce readers to tools and techniques that have been using by QASs. Although we do not use all of them in developing our system, the reader will encounter them somewhere in this thesis. We think that they are beneficial for people interested in natural language processing in general, question answering in particular.

2.2.1 Some basic terminologies

- A **lemma** representing the semantic content of the word. In other words, a lemma is the canonical or base form of the word, such as the form typically found in dictionaries.
- A **part-of-speech tag** representing the abstract lexical category associated with the word.
- **Parsing** means taking an input and producing some sort of linguistic structure for it. One kind of partial parsing is known as chunking.
- **Stemming** is just stripping off word endings to map the word to its root or stem, the main part of the word supplying the main meaning.
- **Lemmatization** is a process of mapping all word surfaces to its root or stem. For example, the lemmatization each of the words *sang*, *sung*, and *sings* produces the word *sing*. In other words, the lemma of these words is the word *sing*.
- **Tokenization** is a task of segmenting running text into words or sentences.

2.2.2 Some natural language processing tools and techniques

Regular Expression

The regular expression language is the standard notation for characterizing text sequences used for specifying text strings in situations such as Web-search and word-processing. It is a powerful tool for pattern-searching. For this purpose, a string is a sequence of alphanumeric characters (letters, numbers, space, tabs, and punctuations). Table 2.1 shows some common alphanumeric characters used in regular expressions.

Table 2.1: Common alphanumeric characters used in regular expression

Character	Description	Example
[ ]	A set of characters	"[a-m]"
\	Signals a special sequence (can also be used to escape special characters)	"\d"
.	Any character (except newline character)	"he..o"
$	Ends with	"world$"
*	Zero or more occurrences	"aix*"
+	One or more occurrences	"aix+"
{}	Exactly the specified number of occurrences	"al{2}"
|	Either or	"falls|stays"
()	Capture and group	
\d	Returns a match where the string contains digits (numbers from 0-9)	"\d"
\D	Returns a match where the string DOES NOT contain digits	"\D"
\s	Returns a match where the string contains a white space character	"\s"
\S	Returns a match where the string DOES NOT contain a white space character	"\S"
\w	Returns a match where the string contains any word characters (characters from a to Z, digits from 0-9, and the underscore _ character)	"\w"
\W	Returns a match where the string DOES NOT contain any word characters	"\W"

Regular expression search requires a pattern that we want to search for, and a corpus of texts to search through. A regular expression search function will search through the corpus and returning all texts that contain the pattern. For example, "$[0-9]+(\.[0-9][0-9])?$" is a regular expression for searching price with or without fractions of dollars.

Part-of-speech Tagging

Part-of-speech tagging is the task of assigning part-of-speech or other syntactic class markers (word class) to each word in a corpus. Parts-of-speech can be divided into two broad supercategories: closed class types and open class types.

Noun, verb, adjective, and adverb are four open class types of words in English. A noun is a name given to the syntactic class in which the words for most people, place, or things occur. Nouns are traditionally grouped into proper nouns and common nouns. Proper nouns such as Einstein, Canada, English are names of specific persons or entities. In written English, proper nouns are usually capitalized. The verb class includes most of the words referring to actions and processes. The adjective class includes many terms that describe properties or qualities.

Some closed class types of words in English include preposition, determiner, pronoun, conjunction, particle, numeral.

Table 2.2 shows the most commonly used Penn Treebank tagsets used in part-of-speech tagging task.

Given a question Q1: "what is the time zone of Salt Lake City?". Its tagged sentence using the Penn Treebank part-of-speech tagsets produced by nltk API in Python is

what/WP is/VBZ the/DT time/NN zone/NN of/IN Salt/NNP Lake/NNP City/NNP?/.

Chunking

Chunking is the process of identifying and classifying the flat, non-overlapping segments of a sentence that constitute the basic non-recursive phrases corresponding to the major content-word parts-of-speech: noun phrases, verb phrases, adjective phrases, and prepositional phrases.

The rules that make up a chunk of grammar using tag patterns to describe sequences of tagged words. A tag pattern is a sequence of part-of-speech tags delimited using angle brackets. For example, in Python we can define chunk tags (CNP-common noun phrase, PNP-proper noun phrase, VP-verb phrase)

Table 2.2: Common part-of-speech tagsets in Penn Treebank

Tag	Description	Example	Tag	Description	Example
CC	Coordin. Conjunction	*and, but, or*	RBR	Adverb, comparative	*faster*
CD	Cardinal number	*one, two, three*	RBS	Adverb, superlative	*fastest*
DT	Determiner	*a, the*	UH	Interjection	*ah, oops*
EX	Existential 'there'	*there*	SYM	Symbol	*+,%, &*
IN	Preposition	*of, in, by*	TO	"to"	*to*
JJ	Adjective	*large*	UH	Interjection	*ah, oops*
JJR	Adj., comparative	*larger*	VB	Verb, base form	*go*
JJS	Adj., superlative	*largest*	VBD	Verb, past tense	*went*
MD	Modal	*can, should*	VBG	Verb, gerund	*going*
NN	Noun, sing. or mass	*car*	VBN	Verb, past participle	*gone*
NNS	Noun, plural	*cars*	VBP	Verb, non-3sg pres	*go*
NNP	Proper noun, singular	*IBM*	VBZ	Verb, 3sg pres	*goes*
NNPS	Proper noun, plural	*Carolinas*	WDT	Wh-determiner	*which, that*
PDT	Predeterminer	*all, both*	WP	Wh-pronoun	*who, what*
POS	Possessive ending	*'s*	WP$	Possessive Wh-	*whose*
RB	Adverb	*fast*	WRB	Wh-adverb	*How, where*

using tag patterns as following

grammar = """

$CNP : < JJ.* > * < NNS? > +$

$PNP : < NNPS? > +$

$VP : < VB.* >< IN >?$

"""

A common noun phrase is previously defined with zero or more optional adjectives followed by one or more common nouns. Running Python nltk chunker produces a result whose tree representation is given in Figure 2.3.

Dependency parser

The dependency parser jointly learns sentence segmentation and labeled dependency parsing, and can optionally learn to merge tokens that had been over-segmented by the tokenizer.

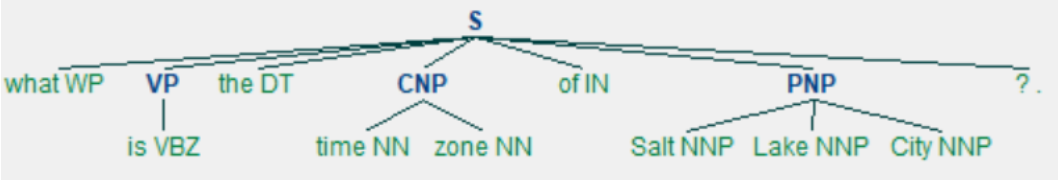


Figure 2.3: Chunking result of the sentence "what is the time zone of Salt Lake City?"

In Figure, the root node, the head of the entire structure, is explicitly marked. Relations among the words are drawn with directed, labeled arcs from heads to dependents. The labels are taken from the Universal Dependency set. Table 2.3 shows some selected grammatical relations from the Universal dependency set [18].

Table 2.3: Some selected grammatical relations from the Universal dependencies

Relation	**Description**	**Relation**	**Description**
acl	clausal modifier of noun	dep	unspecified dependency
advcl	adverbial clause modifier	det	determiner
advmod	adverbial modifier	dobj	direct object
amod	adjectival modifier	iobj	indirect object
appos	appositional modifier	mwe	multi-word expression
case	prepositions, postpositions and other case markers	neg	negation modifier
cc	coordinating conjunction	nmod	nominal modifier
ccomp	clausal complement	nsubj	nominal subject
compound	compound	nsubjpass	passive nominal subject
conj	conjunct	nummod	numeric modifier
csubj	clausal subject	root	root

For example, the Stanford Dependency parser produces a dependency tree of the sentence "what is the time zone of Salt Lake City?" as illustrated in Figure 2.4.

The spaCy dependency parser produces a dependency tree of the same sentence "what is the time zone of Salt Lake City?" as depicted in Figure 2.5.

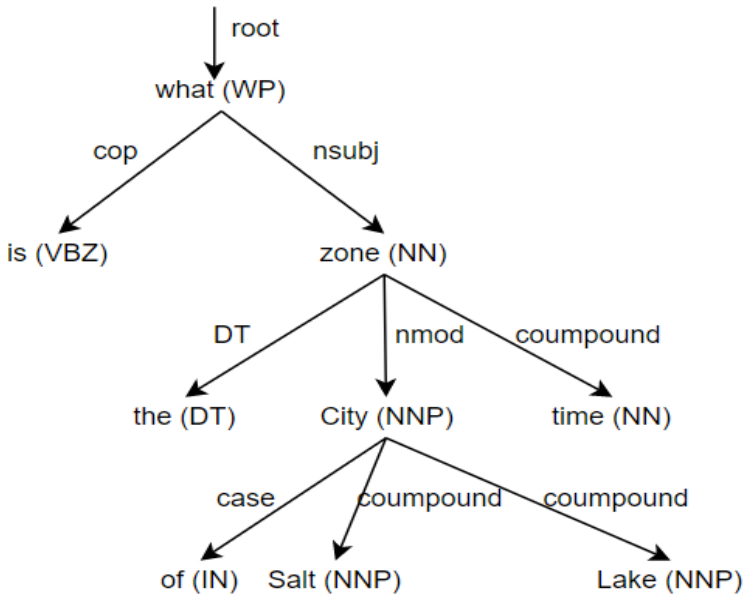


Figure 2.4: The dependency tree of "what is the time zone of Salt Lake City?" produced by Stanford's parser

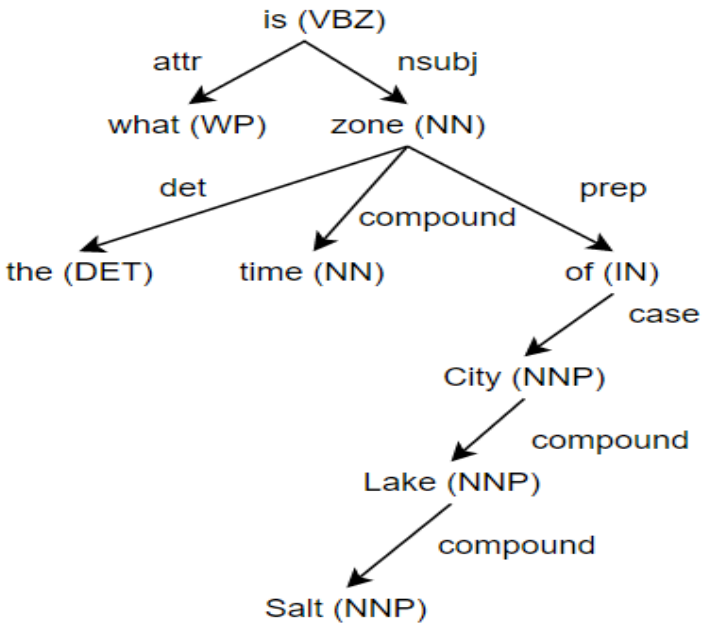


Figure 2.5: The dependency tree of "what is the time zone of Salt Lake City?" produced by spaCy's parser

In the two figures, words are depicted with nodes and relations among the words are illustrated with directed, labeled arcs(arrows) from heads to dependents. The labels are drawn from a fixed inventory of grammatical relations

(see Table 2.3). The root of the tree is explicitly marked. Nodes and arcs are connected creating a upside-down tree. There is a unique path from the root node to each node in the tree.

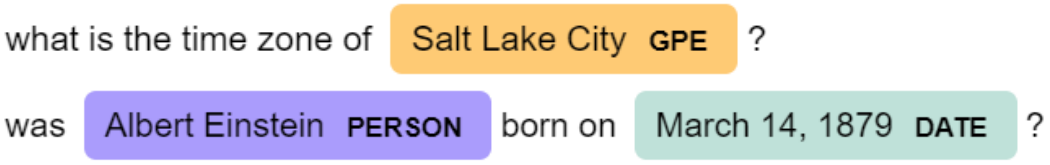
These dependency parser has been widely used by QASs. Some representative systems are NEQA [2], a system by Xu et al. [109], and a system by Hakimov et al. [41].

Named Entity Recognition

Named Entities (NEs) are definite noun phrases that refer to specific types of individuals, such as organizations, persons, dates. Commonly used types of NEs are PERSON, ORG-organization, LOC-location, DATE, GPE-geopolitical.

A named entity recognition system aims at identifying all textual mentions of the named entities by first identifying the boundaries of a NE then identifying its type.

spaCy[10] is a free, open-source library for advanced NLP in Python. Using spaCy's NER on the question Q1 and a new question Q2:"was Albert Einstein born on March 14, 1879?" produces the following results:



WordNet: A lexical database of English

WordNet [33] is a large lexical database of English whose lexicalized concepts are organized by semantic relations (synonymy, antonymy, hyponymy, meronymy, etc.) for nouns, verbs, and adjectives. Semantic relations link the synonym sets.

In WordNet, a form is represented by a string of ASCII characters, and a sense is represented by the set of (one or more) synonyms that have that sense. WordNet contains more than 118,000 different word forms and more than 90,000 different word senses.

[10]https://spacy.io/usage/spacy-101#whats-spacy

Semantic relations in WordNet include the following:

Synonymy is WordNet's basic relation representing a symmetric relation between word forms. The synonyms are grouped into synsets with short definitions and usage examples

Antonymy is also a symmetric semantic relation between word forms, especially important in organizing the meanings of adjectives and adverbs.

Hyponymy (sub-name) and its inverse, hypernymy (super-name), are transitive relations between synsets. Because there is usually only one hypernym, this semantic relation organizes the meanings of nouns into a hierarchical structure.

Meronymy (part-name) and its inverse, holonymy (whole-name), are complex semantic relations. WordNet distinguishes component parts, substantive parts, and member parts.

Troponymy (manner-name) is for verbs what hyponymy is for nouns, although the resulting hierarchies are much shallower.

Entailment relations between verbs.

Many QA systems use WordNet as an additional lexical source, AquaLog [69],BELA [89], Power-Aqua [70], system by Yahya et al. [115], and system by Hakimov et al. [41] for example. However, WordNet only includes lexical paraphrases; it does not include phrasal or syntactically based paraphrase.

2.2.3 String similarity metrics

Distance functions

Distance functions map a pair of strings s and t to a real number r, where a smaller value of r indicates greater similarity between s and t.

Minimum edit distance, named by Wagner and Fischer (1974), between two strings is the minimum number of editing operations (insertion, deletion, substitution) needed to transform one string into another.

Levenshtein distance between two sequences is the simplest weighting factor in which each of the two operations (insertion, deletion) has a cost of 1 while a substitution has a cost of 2(Levenshtein, 1966). This distance measure

has been using by QA systems such as a system by Shekarpour and Auer [90]

Similarity functions

Similarity functions are analogous to Distance functions, except that larger values indicate greater similarity.

Jaro metric Given strings $s = a_1 \dots a_K$ and $t = b_1 \dots b_L$, define a character a_i in s to be common with t there is a $b_j = a_i$ in t such that $i - H \leq j \leq i + H$, where $H = \frac{min(|s|,|t|)}{2}$. Let $s' = a'_1 \dots a'_{K'}$ be the characters in s which are common with t (in the same order they appear in s) and let $t' = b'_1 \dots b'_{L'}$ be analogous; now define a transposition for s', t' to be a position i such that $b'_j \neq a'_i$ Let $T_{s',t'}$ be half the number of transpositions for s' and t'. The Jaro similarity metric for s and t is

$$Jaro(s,t) = \frac{1}{3}.\left(\frac{|s'|}{|s|} + \frac{|t'|}{|t|} + \frac{|s'| - T_{s',t'}}{|s'|}\right) \tag{2.1}$$

A variant of this due to Winkler (1999) also uses the length P of the longest common prefix of s and t. Letting $P = max(P, 4)$ we define

$$Jaro - Winkler(s,t) = Jaro(s,t) + \frac{P'}{10}.(1 - Jaro(s,t)) \tag{2.2}$$

Jaccard similarity between the word sets S and T is simply $\frac{S \cap T}{S \cup T}$.

term frequency–inverse document frequency (tf-idf) can be defined as

$$tf - idf(S,T) = \sum_{w \in S \cap T} V(w,S).V(w,T) \tag{2.3}$$

where

$$V(w,S) = \frac{V'(w,S)}{\sqrt{\sum_{w'} V'(w,S)^2}}$$

and

$$V'(w,S) = log(TF_{w,S} + 1).log(IDF_w)$$

where $TF_{w,S}$ is the frequency of word w in S, N is the size of the "corpus",IDF_w is the inverse of the fraction of names in the corpus that contain w.

Vector-based distance

Euclidean distance: given data points in R^d, the Euclidean distance metric is

$$Euclidean(x, y) = \sqrt{\sum_{i=1}^{d}(x_i - y_i)^2} \tag{2.4}$$

cosine similarity: Given two non-zero vectors x and y, their cosine similarity is calculated as

$$cosine(x, y) = \frac{\sum_{i=1}^{d}(x_i.y_i)}{\sqrt{\sum_{i=1}^{d}(x_i)^2}.\sqrt{\sum_{i=1}^{d}(y_i)^2}} \tag{2.5}$$

2.3 Fuzzy sets

Fuzzy sets [118] were introduced by Dr. Zadeh in a seminal paper published in 1965. A fuzzy set contains its elements whose degrees of membership to the set are not a matter of affirmation or denial, but rather a matter of degree. A membership degree is a value assigned to an element of A based on the similarity or compatibility with the concept represented by the fuzzy set. A larger value denotes a higher degree of set membership [84]. The formal definition of A as follows:

$$A : X \to [0, 1] \tag{2.6}$$

It can also be defined as a set of pairs of the form:

$$A = \{(x, \mu_A(x))|x \in X\} \tag{2.7}$$

where $\mu_A(x)$ is a degree of membership of x in the fuzzy set A.

The ability of expressing gradual transitions from membership to nonmembership and vice versa makes fuzzy sets highly applicable, one of them is in representing vague concepts expressed in natural language [84].

Height of a fuzzy set A, denoted by hgt(A), is determined as

$$hgt(A) = sup(A(x)) \ for \ x \in X \tag{2.8}$$

By determining the height of the fuzzy set, we identify elements with the highest membership degree. They are consider the representatives of the concept represented by A.

The fuzzy set A is normal if $hgt(A) = 1$.

A set consisting of the elements of the universe whose membership values to a fuzzy set A are equal to or exceed a certain threshold level α ($\alpha \in [0, 1]$) is called the $\alpha - cut$ of the fuzzy set A, and is denoted by A_α. It is formally defined as

$$A_\alpha = \{x \in X | A(x) \geq \alpha\} \tag{2.9}$$

A strong $\alpha - cut$ of a fuzzy set A contains all X's elements whose membership values to A are greater than α. It is denoted A_α^+ and formally defined as

$$A_\alpha^+ = \{x \in X | A(x) > \alpha\} \tag{2.10}$$

The support of a fuzzy set A, denoted by Supp(A), is defined as a set containing all nonzero membership elements of X. In other words, it is defined as

$$Supp(A) = \{x \in X | A(x) > 0\} \tag{2.11}$$

The core of a fuzzy set A, denoted by Core(A), is defined as a set containing all X's elements whose membership grades equal to 1. In other words, it is defined as

$$Core(A) = \{x \in X | A(x) = 1\} \tag{2.12}$$

The cardinality of a fuzzy set A defined in a finite or countable universe X, denoted by $Card(A)$, is expressed as following sum:

$$Card(A) = \sum_{x \in X} A(x) \tag{2.13}$$

2.4 Ordered Weighted Aggregation Operator

Ordered Weighted Aggregation (OWA) [114] is one of most widely used numeric data aggregating operators. In the simplest possible statement, this operator is a weighted sum over ordered pieces of information. In a formal representation, the OWA operator, defined on the unit interval I and having dimension n, is a mapping

$OWA : I^n \longrightarrow I$ such that

$$OWA(a_1, a_2, ..., a_n) = \sum_{j=1}^{n} w_j * b_j \tag{2.14}$$

where b_j is the j^{th} largest of the a_i's. $W = \{w_1, w_2, ..., w_n\}$ is a weighting vector such that $0 \leq w_j \leq 1$ and $\sum_{j=1}^{n} w_j = 1$.

To obtain a weighting vector W associated with an OWA, Dr. Yager introduced families or regular increasing monotone (RIM) quantifier Q. A fuzzy subset Q represents a RIM quantifier if

1) $Q(0) = 0$;

2) $Q(1) = 1$;

3) if $r_1 > r_2$ then $Q(r_1) > Q(r_2)$.

Assuming that a RIM quantifier Q, the weighting vector W can be determined such that for $j = 1$ to n:

$$w_j = Q(\frac{j}{n}) - Q(\frac{j-1}{n}) \tag{2.15}$$

2.5 Linguistic Summarization

Dr. Yager [110] introduced linguistic summarization that summarizes observations of the property V for elements of the set $Y = \{y_1, y_2, ..., y_n\}$ that are

represented as a collection of values $D = \{v_1, v_2, ..., v_n\}$, where $v_i = V(y_i)$. A linguistic summary requires:

1) a summarizer, S: a fuzzy set characterized by its membership function $\mu_S(v)$, $\forall v \in D$;

2) a quantity in agreement, Q: a fuzzy linguistic quantifier being a fuzzy set, characterized by $\mu_Q(x)$, $x \in [0, 1]$;

3) a measure of validity or truth of the summary – T – can be calculated according to Zadeh's (1983) calculus of linguistically quantified propositions as.

$$T = \mu_Q(\frac{1}{n}\sum_{i=1}^{n}\mu_S(v_i)) \tag{2.16}$$

2.6 Question answering systems

A question-answering (QA) system is a computer system that can automatically answer questions posed by humans in natural languages. Over 60 years, many systems have been developed, from domain-specific systems such as BASEBALL [40] and LUNAR [103] to open-domain complex systems such as IBM Watson [36], Google Assistant, Apple's Siri, Amazon's Alexa.

Many approaches have been using in developing QA systems. The most widely used ones are Information Retrieval (IR) based, template-based, and most recently neural networks based. The detail of each approach together with representative systems that have been adopting the approach will be presented in Section 3.7.

Chapter 3

Human-centric Question-Answering System

Question answering is a research area applying information retrieval, natural language processing, and machine learning techniques in building systems that automatically answer questions posed by humans in a natural language. Many question answering systems have been developing so far. However, most of them are only able to answer questions that have a short answers. In this chapter, we introduce our human-centric system. We will provide the system's overview. Next, we present its main components in more detail and some extensions that make the system distinctive. We also make a short survey on approaches adopted by such systems.

3.1 System Overview

A 'traditional' Question-Answering system (QAS) could be, in a simplified form, presented as shown in Figure 3.1. Once a user asks a question, the system queries a knowledge base – could be a structured database as in the case of WDAqua-core0 [22], or unstructured set of documents as in the case of the Wikipedia DrQA [13] – and an answer is 'displayed' to the user.

In comparison with a 'traditional' QAS, our human-centric one – called *Linguistic Term Question-Answer system (LingTeQA)* – will be equipped with a number of extensions as illustrated in Figure 3.2

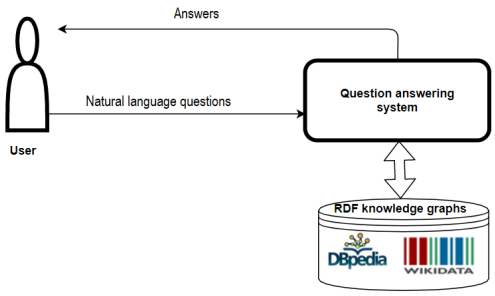


Figure 3.1: A 'traditional' question answering system

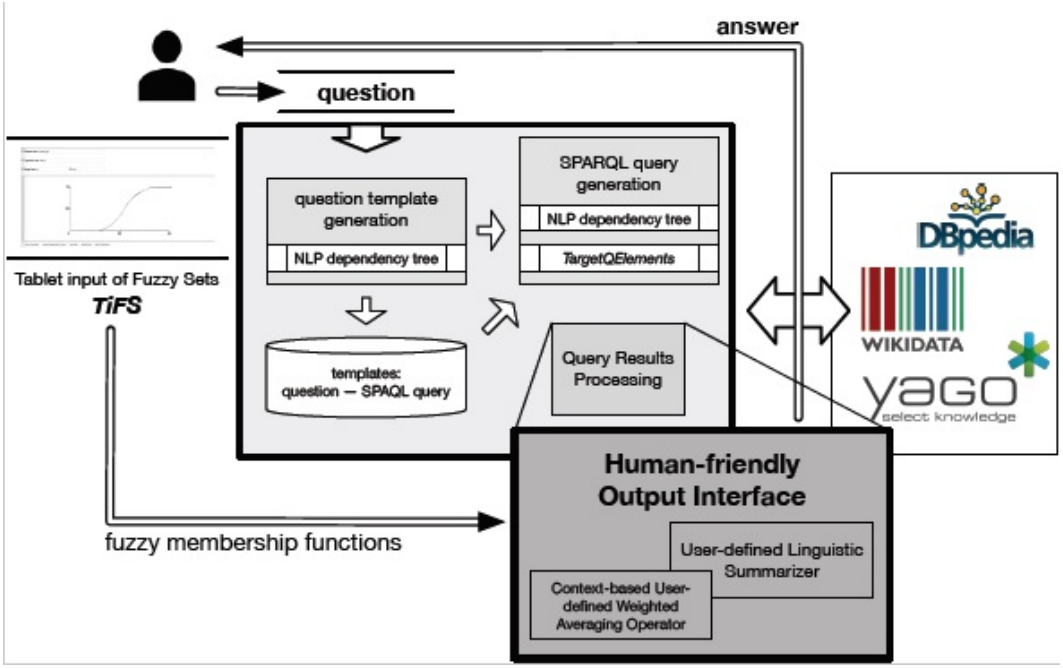


Figure 3.2: *LingTeQA*: A Human-centric Question Answering system

The system – *LingTeQA* – will allow for using imprecise concepts, asking questions that required additional processing, and providing more summary-like answers. To do so, we aim at developing algorithms, methods, and tools necessary for doing following tasks: 1) template generation from pairs of question and SPARQL query; 2) question to query translation based on generated templates; 3) data collection from Knowledge Graphs (KGs) by executing generated queries; 4) data fusion on answers collected from multiple different KGs; 5) construction of definitions of user-based linguistic terms and quantifiers (if

any) so questions that contain such terms can be answered accordingly to their understanding of the terms; 6) data summarization in forms of linguistic summaries and aggregated values (if suitable).

3.2 Question Representation

We use a so-called hierarchical structure, namely a dependency tree, for representing English questions. A tree is a set of connected labeled nodes, each reachable via a unique path from a distinguished root node.

3.2.1 Phrasal Dependency Definition

A phrasal dependency tree of a sentence is a directed graph representation, in which phrases in the sentence are nodes and grammatical relations are labeled edges in the graph. A phrase, also called a constituent, is a set of words that act together as a unit. Every phrase has a head that determines the category of the phrase. So if the head is a noun, then our phrase is a noun phrase (NP), abbreviated NP-common noun/PN-proper noun. If the head is a verb then the phrase is a verb phrase (VP), etc.

3.2.2 Phrasal Dependency Tree Generation

We perform the following steps to generate a phrasal dependency tree for a user's question.

Step 1. By applying the *spaCy* dependency parser [46], we obtain typed dependencies that are triples of a relation between pairs of words of the question. For example, given the question "what is the time zone of Salt Lake City?" the *spaCy* parser produces typed dependencies whose graphical representation is drawn as a tree in Figure 3.3 (with part-of-speech tags presented in parenthesis).

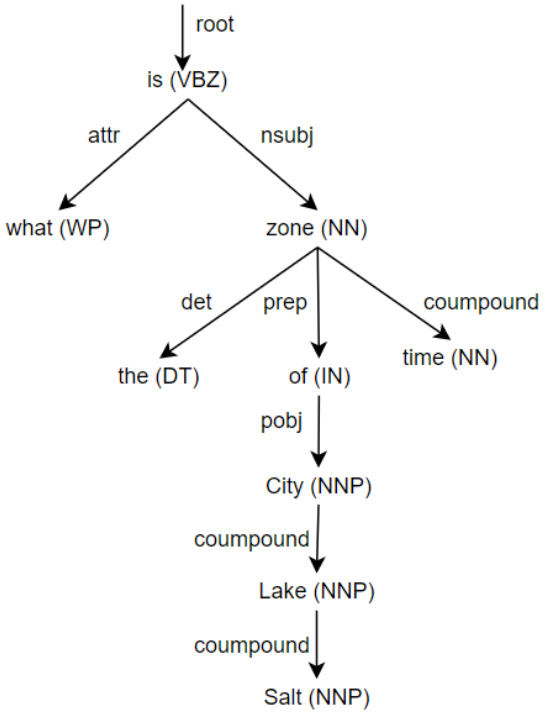


Figure 3.3: Word-level dependency tree of the question "what is the time zone of Salt Lake City?" produced by *spaCy*

Step 2. By modifying the *spaCy* original dependency tree (word-level tree), using specialized heuristics so that it becomes a phrase-level tree. In particular, words that involve a multiword-expression relation such as 'compound', 'mwe' are combined to form phrases. Part-of-speech tags of phrases are also renamed to indicate common noun phrase-(NP), proper noun phrase-(PN), verb phrase-(VP). The phrase-level dependency tree for the question is illustrated in Figure 3.4.

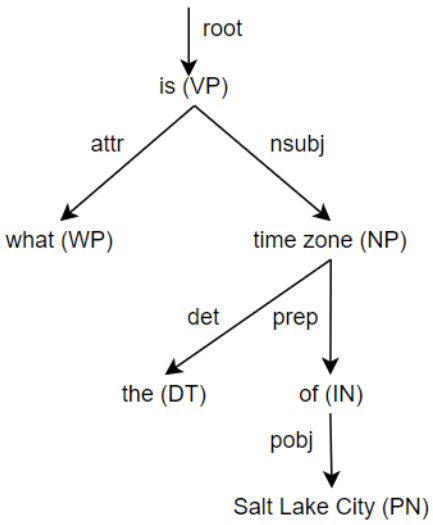


Figure 3.4: Phrase-level dependency tree of the question "what is the time zone of Salt Lake City?"

3.3 Template Generation Process

The template generator is responsible for generating a pair of ⟨question template, query template⟩–⟨Qst, Qrt⟩ from a given pair ⟨natural language question, SPARQL query⟩–⟨Qs, Qr⟩ for a given, specified by the user, Knowledge Graph (K). The template generator is composed of a question template generator and a query template generator.

3.3.1 Question Template Generator

A question template (Qst) of a given question (Qs) is generated by traversing the phrasal dependency tree of the question and printing out an accessing path of every visited node and associated part-of-speech tag of the node. An accessing path of a node in the tree is a sequence of connections starting from the root following by edges(arcs) that lead to the node. Figure 3.5 depicts the question template generation process.

Figure 3.6 shows an example of question template generation of the question "what is the time zone of Salt Lake City?" whose phrasal-level dependency tree depicted in Figure 3.4.

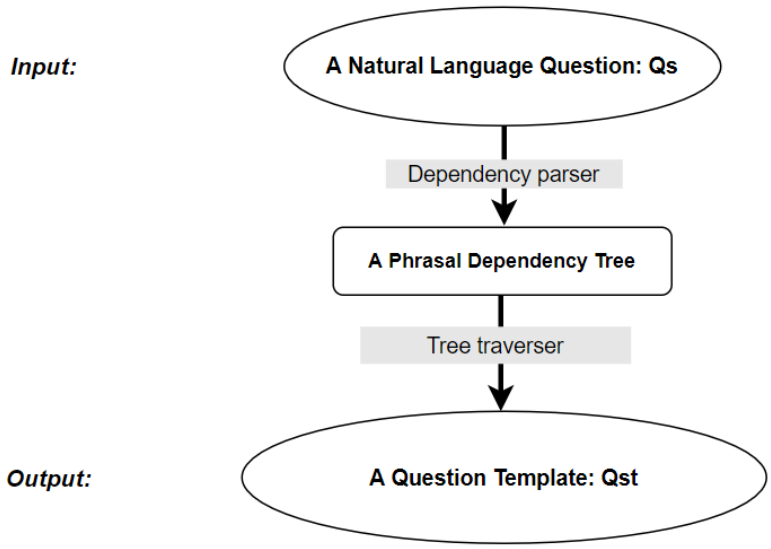


Figure 3.5: Question template generation process

3.3.2 Query Template Generator

Given a pair ⟨natural language question-SPARQL query⟩ and a knowledge graph (K), the query template generator is in charge of generating a SPARQL query template (Qrt) from the SPARQL query. It is done by replacing specific elements of the SPARQL query (Qr) with placeholders that express the category of the query's elements and the associations between them and phrases reside in nodes of the dependency tree of the question (Qs). The Figure 3.7 presents the query template generation process.

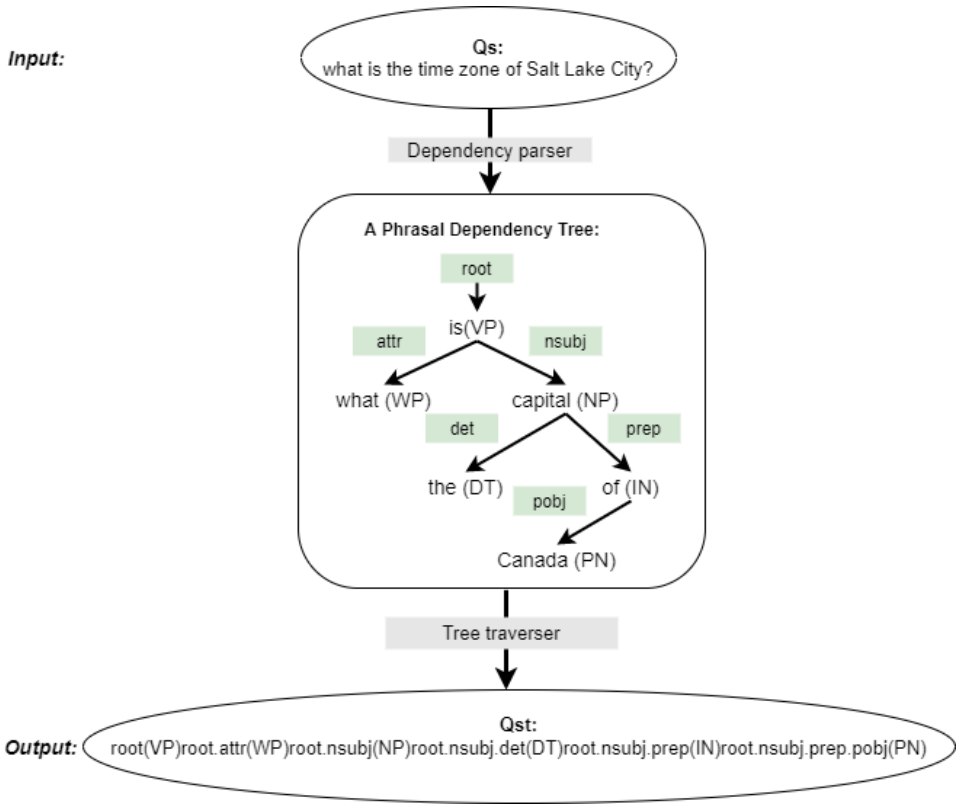


Figure 3.6: Question template generation process with an illustrative question

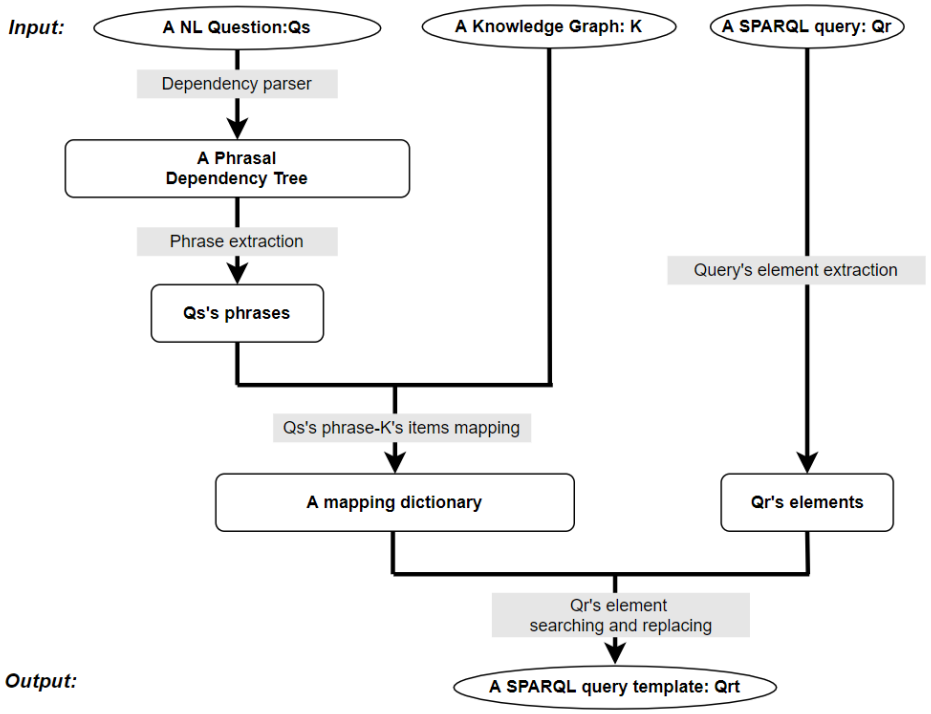


Figure 3.7: Query template generation process

The whole process of generating a pait ⟨question template, query template⟩– ⟨Qst, Qrt⟩ performed on an illustrative example is shown in Figure 3.8. We explain the process of mapping phrases form natural language questions into a knowledge graph's semantic items (URIs) in the next subsection.

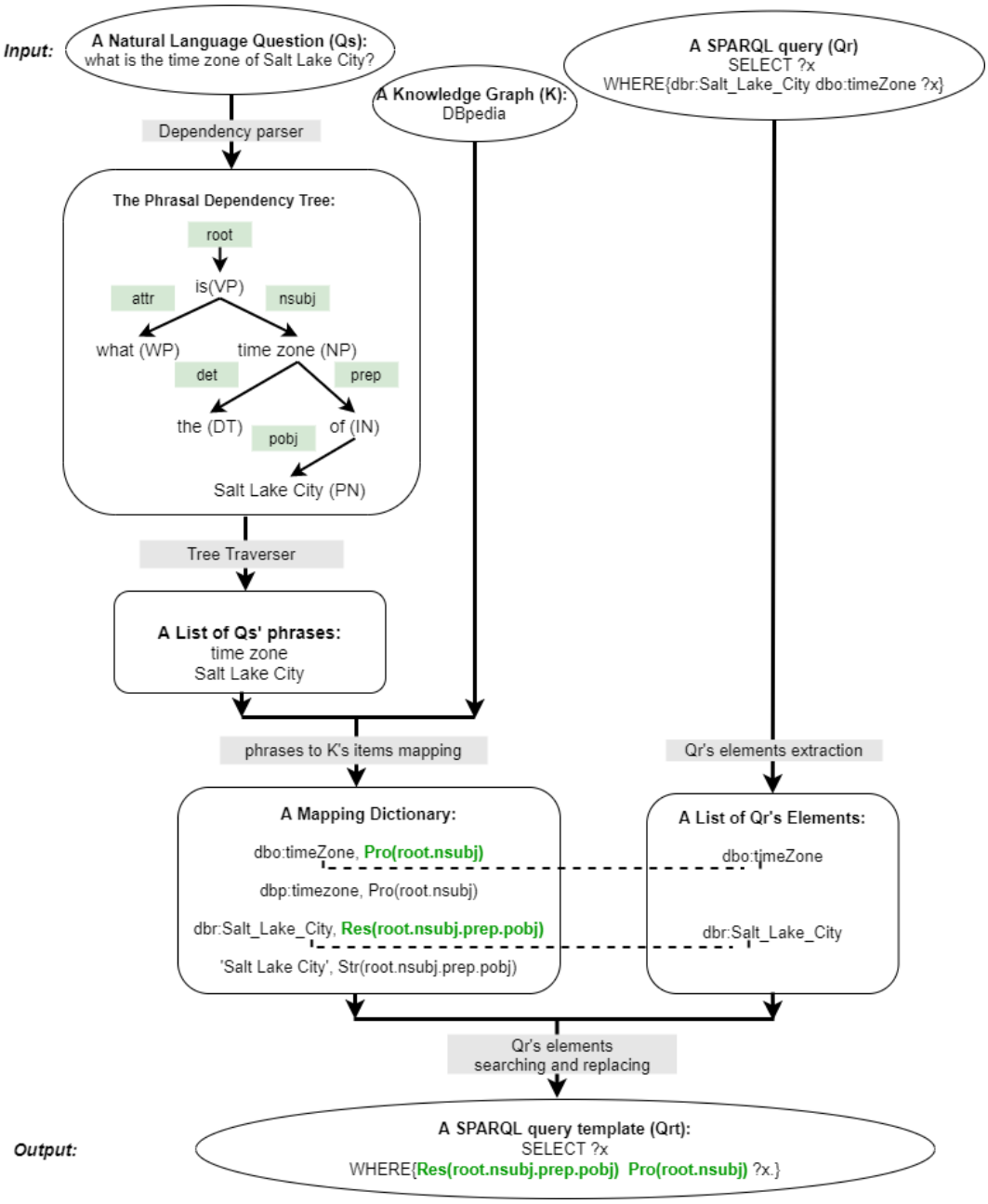


Figure 3.8: Query template generation process with an illustrative question

3.3.3 Mapping natural language expressions into Knowledge Graph's semantic items

Mapping a natural language expression (phrase) found in the user's question into a target Knowledge Graph's semantic items, aka URIs, (classes, entities, properties) is done by matching the expression with URIs' labels. However, the variability of natural language phrases and terms used in the Knowledge Graph creates differences between the phrases and terms. Usually, external sources such as WordNet and lexicons, are used to find related phrases (synonyms, different word surfaces,...) to narrow the differences. AquaLog [69],PowerAqua [70], for instance, use WordNet [33] for finding synonyms, antonyms. NEQA [2] uses manually created lexicons such as predicate lexicon and class lexicon.

In *LingTeQA*, we use items and properties labels from WordNet, Wikidata, and a small manually created lexicon created by us, to obtain additional phrases. The original phrases and the additional phrases are then mapped into a target Knowledge Graph's semantic items with respect to their lexical categories. In particular:

- A verb phrase identified by a **'VP' part-of-speech (POS)** will be treated in the following ways. 1) It will be converted into a noun phrase using WordNet. For example, 'wrote' will be converted into 'writer'. Noun versions of some verbs will also be combined with question words creating noun phrases such as 'birth place', 'birth date'. 2) The obtained noun phrase will be used to find synonymous phrases using WordNet. For instance, by using 'writer' the system finds 'author' as a synonym; 3) The obtained phrases will be used to find synonymous expressions using 'alt' labels in Wikidata. For example, by using 'author' to look up synonymous expressions in Wikidata, the system finds 'creator'. The original verb phrase and additional noun phrases obtained via the three aforementioned steps are mapped into properties of a target knowledge graph using a label matching procedure. For example, by using label matching between DBpedia's properties and obtained phrases, *LingTeQA* ob-

tains *'dbp:writer', 'dbo:writer', 'dbp:author', 'dbo:author', 'dbp:creator', 'dbo:creator'*.

- A common noun phrase identified by an **'NP' POS** will be undertaken the following processing. 1) It will be used to find its canonical form (lemma) using WordNet. For example, 'writer' is a lemma of 'writers'; 2) The obtained noun phrase will be mapped into a target Knowledge Graph class. For example, the system obtains *'dbo:Writer* class when mapping 'writer' into a DBpedia's class. The obtained noun 'writer' is also used to find synonymous phrases using WordNet and Wikidata. As mentioned above the system has found 'author' and 'creator' as synonyms of 'writer'. The obtained phrases will be mapped into DBpedia's properties such as *'dbp:writer', 'dbo:writer', 'dbp:author', 'dbo:author', 'dbp:creator', 'dbo:creator'*.

- A proper noun phrase identified by a **'PN' POS** will be processed by the following steps. 1) It will be mapped into a target Knowledge Graph entity (individual) using Entity lookup Application Programming Interfaces (APIs) together with querying the Knowledge Graph using its SPARQL Endpoint. *LingTeQA* uses WikiData's API, DBpedia's API to obtain a list of entities. In case an empty result list is obtained, for example from DBpedia, the system uses the API of Wikidata, and then it uses the property *owl:sameAs* to find entities in DBpedia, and vice versa. Next, the system will select among obtained entities those whose label is best matched with the proper noun. For example, the system obtains *wd:Q16, wd:Q1121436, wd:Q2569593, wd:Q257304, ...* when using Wikidata's API with the keyword 'Canada'. Among them, the system selects *wd:Q16* because its label best matches the keyword. By using the *owl:sameAs* property the system finds *'dbr:Canada* as a targeted resource in DBpedia. The proper noun is also mapped into a string constant.

- An adjective identified by an **'AJ' POS** will be processed accordingly –

if the phrase belongs to a nationality/language lexicon, a corresponding country's name will be retrieved, and the retrieved proper noun will be processed similarly to a proper noun mentioned above. WordNet is also used to find a corresponding nouns from adjectives. For example, the adjective 'high' when using WordNet produces a noun 'height'. WordNet also provides attributes of some adjectives. For instance, attributes of 'high' are 'level','degree', 'grade'. Attributes of the adjective 'female' are 'gender', 'sex', 'sexuality'. Obtained noun phrases are then processed similarly to common nouns. A comparative adjective or adverb will be mapped into comparative signs using a manual lexicon as well. For instance, 'higher/more' is translated into the sign ' $>'$.

- A cardinal phrase, identified by a **'CD' POS** will be converted into a number using our own defined function.

There are expressions that do not correspond to any vocabulary element. Examples are quantifiers like 'the most', comparative expressions like 'more than', cardinals, and superlatives. These expressions correspond to aggregation operations in SPARQL, such as filtering, ordering, and limits. We use a fixed, dataset-independent meaning lexicon for mapping them into query elements.

When the question's phrase is mapped into a query element such as a KG resource, a constant, or a comparative sign, a placeholder is created to encode the mapping process. A placeholder is a string in a function-like format where the function name is a three-letter string that indicates the type of the target element, for instance, 'Cla' means class, 'Res' means resource, 'Pro' means property. The function argument is a string that specifies the position of the node of the phrasal dependency tree where the associated phrase stored. For example 'Res(root.nsubj.prep.pobj)' is a placeholder in a query template (Qrt) indicating that a phrase that resides at the node determined by the accessing path 'root.nsubj.prep.pobj' must be mapped into a resource of a target Knowledge Graph during the process of constructing an executable query from the query template.

3.4 Answering Question Process

During answering questions, the system – *LingTeQA* – processes a question asked by a user. First, the system analyzes it using the procedure applied during the template generation process and creates a question template. This template is used as a key to retrieve a corresponding SPARQL template from the template repository. The retrieved SPARQL template is then populated with specific information extracted from the asked question and semantic items obtained from a mapping process (see Section 3.3.3). Next, it detects linguistic term(s). If there is no linguistic term in the question, the system invokes a procedure for answering regular questions, otherwise, it invokes a process for answering questions with linguistic terms (see Section 4.4). Figure 3.9 provides an overview of the answering question process.

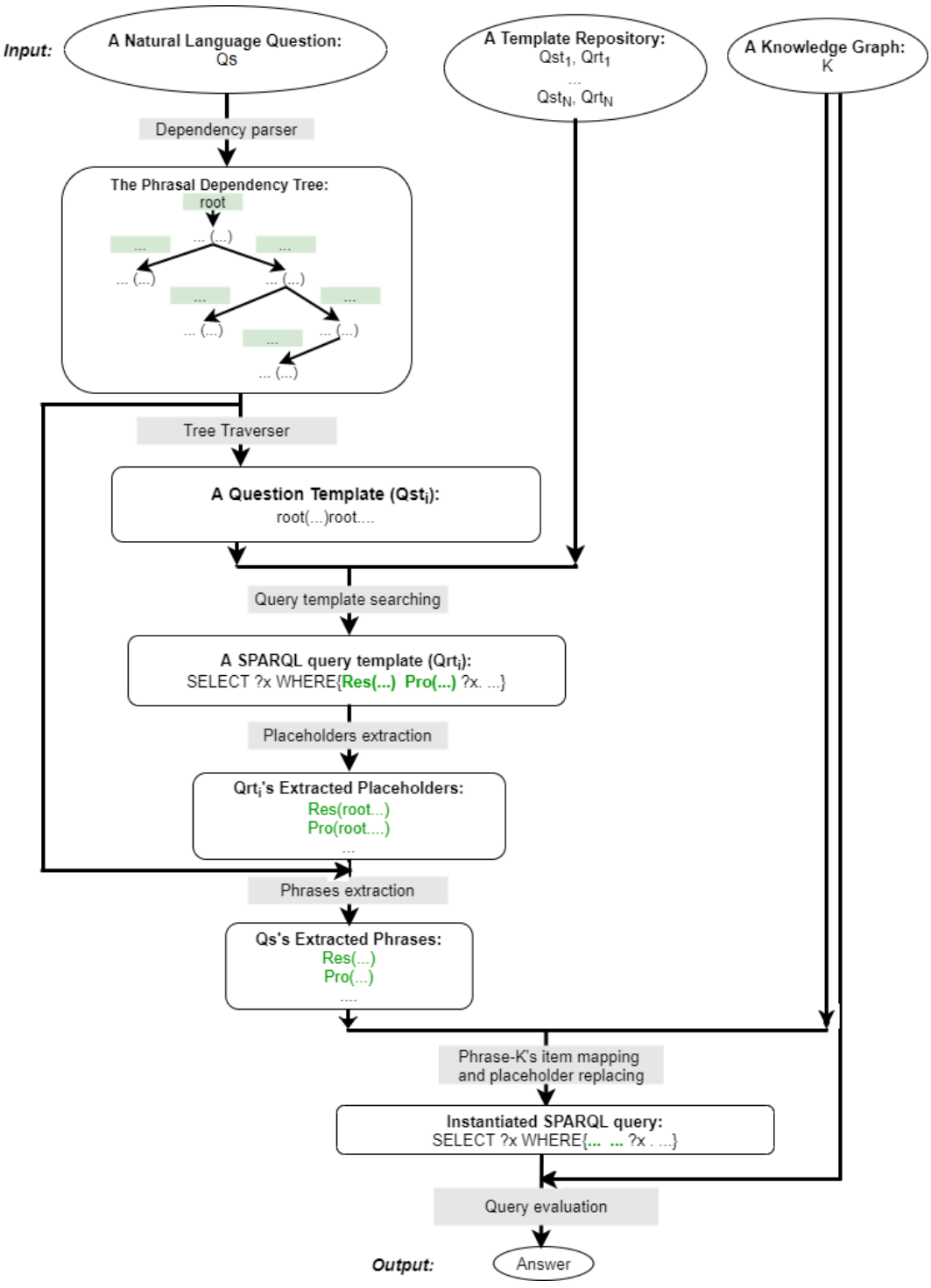


Figure 3.9: *LingTeQA*'s process of answering question without linguistic terms

To illustrate the question answering procedure of *LingTeQA*, let us take an example question "what is the capital of Canada?". Figure 3.10 shows the

process.

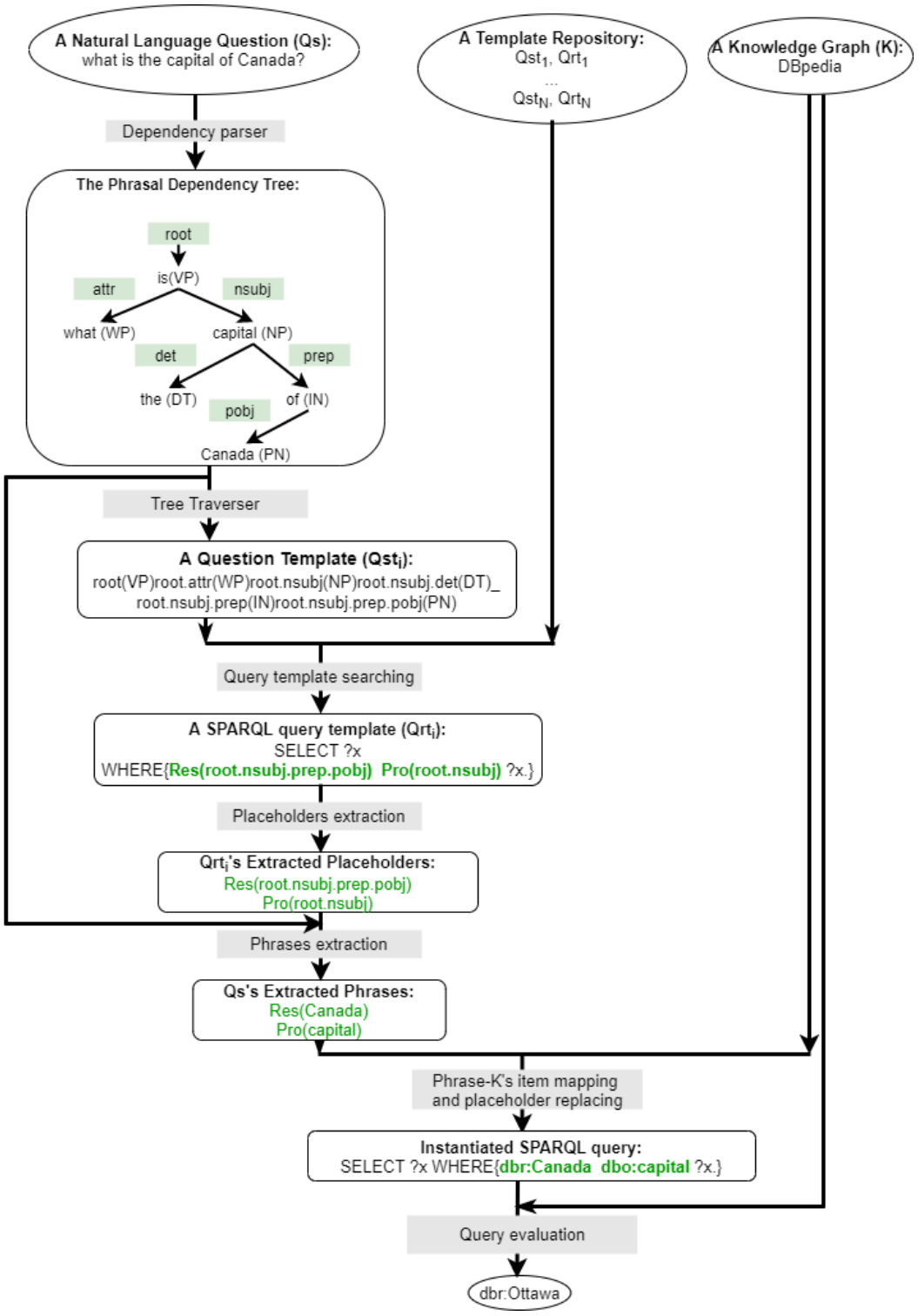


Figure 3.10: *LingTeQA*'s process of answering an illustrative question

3.5 Evaluation of Human-centric QA System on regular questions

3.5.1 Evaluation Datasets

To assess the performance of QA systems a number of datasets have been proposed. Here, we introduce some of them.

Free917 Free917 is a dataset created by Cai and Yates [12] consisting of 917 questions taken from 81 domains of the Freebase database involving 635 relations, annotated with lambda calculus forms.

WebQuestion Because Free917 requires logical forms, it is difficult to scale up due to the required expertise of annotating logical forms. Berant et al. [7] created a new dataset of 5,810 question-answer pairs obtained from non-experts and named it WEBQUESTIONS. Different from FREE917 which starts from Freebase properties and solicits questions about these properties, WEBQUESTIONS starts from questions completely independent of Freebase, and therefore the questions tend to be more natural and varied.

SimpleQuestion SimpleQuestions was the first large-scale dataset of questions and answers based on Freebase contains 108,442 questions written by human annotators so that they are different as much as possible if annotators encounter multiple facts with similar relationship. [10]

ComplexWebQuestion ComplexWebQuestion [91] is a dataset containing 34,689 question-answer pairs of complex questions for evaluating QA systems requiring reasoning over multiple pieces of information. Its authors took queries from WebQuestion to automatically generate more complex ones by adding function composition, conjunctions, superlatives, or comparatives. The obtained queries were then executed against Freebase providing people at Amazon Mechanical Turk answers to generate corresponding natural language questions.

Question Answering over Linked Data

Question Answering over Linked Data (QALD)[1] is a series of evaluation campaigns on question answering over linked data. Since 2011 it has provided up-to-date benchmarks for assessing and comparing state-of-the-art systems that mediate between a user and RDF data. The current QALD challenge is the QALD-9th containing two tasks: Task 1-Multilingual question answering over DBpedia, and Task 2-English question answering over Wikidata.

LC-QuAD is a Large-Scale Complex Question Answering Dataset. Its original version comprised 5000 questions and corresponding SPARQL queries over the DBpedia. The dataset included complex questions exhibiting large syntactic and structural variations i.e. questions in which the intended SPARQL query does not consist of a single triple pattern [3]. In particular, among them, only 18% are simple questions, and the remaining questions require queries either involving more than one triple, or COUNT/ASK keyword, or both. However, there are no queries with OPTIONAL or UNION keywords in the dataset. Also, there are no conditional aggregates in the query head. [3].

LC-QuAD 2.0 consists of 30,000 pairs of questions and its corresponding SPARQL query. The dataset is compatible with both Wikidata and DBpedia 2018 Knowledge Graphs [27].

3.5.2 Experiment Setting and Results

To evaluate LingTeQA's performance, we did not choose Free917, WebQuestion, or ComplexWebQuestion because they are Freebase oriented datasets and Freebase had been shut down by Google since 2014. We used QALD-9th[2] with 408 training questions and 150 testing questions against DBpedia.

We generated a template repository of 107 pairs <*question template – SPARQL query template*> from the 408 pairs of <*question-SPARQL query*> in the training dataset. The constructed template repository was used to answer 150 questions over DBpedia, and 100 questions over Wikidata (based on QALD-7th challenge).

[1] http://qald.aksw.org/

[2] https://2018.nliwod.org/challenge

Table 3.1: Results of QA systems answering QALD-9 (DBpedia)

QA system	macroP	macroR	macroF1
WDAqua	0.049	0.053	0.050
ganswer2	0.097	0.116	0.098
TeBaQA	0.129	0.134	0.130
LingTeQA	**0.242**	**0.295**	**0.246**
Elon	0.261	0.267	0.250
gAnswer (WS)	0.293	0.327	0.298

Table 3.2: LingTeQA results: QALD-9 DBpedia & QALD-7 Wikidata

Dataset	microP	microR	microF1
DBpedia	0.526	0.642	0.535
Wikidata	0.634	0.735	0.642

LingTeQA was able to answer 69 out of 150 over DBpedia, and 55 out of 100 questions over Wikidata. We used precision (P), recall (R), and F1 for comparing our system with QASs that participated in the QALD challenge.

Table 3.1 shows performance of *LingTeQA* in comparison with the performance of systems participated in the 9th QALD challenge [99]. The macro precision, recall, and F1 are calculated based on 150 questions. When we evaluated our QA system on questions over the Wikidata dataset from QALD-7, we obtained the values 0.336, 0.390, and 0.340, respectively. The analysis of obtained results has led us to an interesting observation that some questions could not 'find' entries in the system template repository. Therefore, we have performed another experiment in which we focus on answerable questions, i.e., questions whose syntactic structure identified by *LingTeQA*. The micro precision, recall, and F1 of that experiment for both DBpedia and Wikidata are presented in Table 3.2.

3.5.3 Discussion

There are several questions whose syntactic structures do not match the structures of questions in the training dataset. In particular, they are 81/150 questions over DBpedia and 45/100 questions over Wikidata. For example, "Which American presidents were in office during the Vietnam War?" and "How many gold medals did Michael Phelps win at the 2008 Olympics?". As a result, *LingTeQA* could not find corresponding query templates.

LingTeQA failed to answer some questions whose syntactic structures are even known to it because the ground-truth queries of these questions require very specific semantic items. For example, the query

$SELECT\ ?answer$

$WHERE\ \{dbr{:}Chiemsee\ dbp{:}depth\ ?answer.\}$

was constructed to answer the question "how deep is Lake Chiemsee?" as many people may expect. However, the correct query must be

$SELECT\ DISTINCT\ ?n$

$WHERE\ \{dbr{:}Chiemsee\ dbo{:}maximumDepth\ ?n.\}$

The same situation is applied to other questions such as "What is Batman's real name?".

There are also some questions, to which our system responded with more answers when compared with the reference answers because 1) system-constructed queries are executed against the current DBpedia version instead of the DBpedia 2016-10; 2) our system constructs more properties representing a relationship between two entities than queries in the testing set. As a result, this leads to a decrease in precision but an increase in recall. For instance, our system constructed the query

$SELECT\ DISTINCT\ ?a$

$WHERE\ \{?a\ rdf{:}type\ dbo{:}Language; dbo{:}spokenIn\ dbr{:}Pakistan\}$

to answer the question "What languages are spoken in Pakistan?", while the required query should be

$SELECT\ DISTINCT\ ?uri$

$WHERE\ \{dbr{:}Pakistan\ dbo{:}language\ ?uri\}$

We argue that the structure of the required query is not generic enough for answering the same type of questions.

The performance of *LingTeQA* has also been degraded because it has failed in mapping from natural language phrases into DBpedia's items. For example, the system wrongly has mapped "Indigo" into dbr:Inigo_Jones instead of dbr:Indigo. The mapping from a natural language expression into a DBpedia's property is even more challenging. For instance, to retrieve the mayor of a city from DBpedia, the corresponding property is dbo:mayor in the case of Lyon, dbo:leader in the case of Berlin, and dbo:leaderName in the case of Tel Aviv. In many cases, the mapping process requires not only a constituent of a question but the question as a whole. That is the case of the question "When did Finland join the EU?". *LingTeQA* has failed to map "join" to dbp:accessioneudate.

In addition, some questions are challenging to any existing QA system because they require reasoning related to temporal data. For example, the questions "Which American presidents were in office during the Vietnam War?" and "Give me all American presidents of the last 20 years."

To deal with questions we have not found templates for, we have adopted a process called paraphrasing. The more details related to it and its usefulness are included in the following section – Section 3.6.

3.6 Paraphrasing

3.6.1 Motivation

Many studies showed that state-of-the-art QASs are very sensitive to variations in the way questions are worded [78][24], and *LingTeQA* is no exception.

The performance of many Question-Answering systems can improve via adopting paraphrasing as their first processing step. For example, Duboue and Chu-Carroll [29] reported that replacing a question with a more felicitously worded question can potentially result in a 35% performance enhancement. Similarly, by supplying their QAS with a question and its paraphrases, Dong et al. [24] reported that the system consistently improves performance, achiev-

ing competitive results despite the use of simple question answering models. Paraphrasing can generate an array of lexically and structurally distinct rewordings, some of them may better match the processing capabilities of the system than the original questions. As a result, the question is more likely answerable and correctly answered [29].

The idea of using a template-based paraphraser as part of a natural language question-answering system has been around since the 1970s when such systems as PLANES developed by Waltz et al. and RENDEZVOUS proposed by Codd et al. had used templates to form paraphrases. It has been done by filling empty slots in the pattern with information from the user's question [58]. Recently, Fader et al. [31] [32] have introduced PARALEX, a paraphrased-based system that can answer open-domain questions with single-relation queries.

Another widely-used technique for paraphrasing uses multiple parallel corpora and machine translation [5][4][29][73]. Ganitkevitch et al. [38] have combined several English-to-foreign bitext corpora to extract PPDB:Eng, a large ranked ParaPhrase DataBase. The PPDB:Eng contains over 220 million paraphrase pairs, consisting of 73 million phrasal and 8 million lexical paraphrases, as well as 140 million paraphrase patterns. An improved version of PPDB has been done by Pavlick et al. [83] and Fujita and Isabelle [37]. Narayan et al. [78] have trained their model on the PARALEX corpus using the lexical and phrasal rules from the PPDB, and generated both lexically and syntactically diverse paraphrases.

Recently, thanks to the advances in deep learning algorithms leading to the construction of several deep neural network models – Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), Long Short Term Memory (LSTM), with some of them focused on NLP tasks. One of the most interesting techniques of deep learning is transfer learning. In this technique, a model is first pre-trained on a data-rich task before being fine-tuned on a similar task. This has emerged as a powerful technique in natural language processing [85]. Text-To-Text Transfer Transformer (T5) model developed by Google's research team treats every text processing problem as a "text-to-text" problem, i.e., it

takes a text as an input and produces a new text as the output.

3.6.2 Paraphrasing and Human-centric QA System

In *LingTeQA*, because a template that is generated from a given pair of $< question - query >$ can not be used to answers paraphrases of the question, we use a *T5 transformer*[3] to generate up to ten paraphrases of the question. As a result, up to 10 new pairs– $< paraphrasedquestion, query >$– are generated. We also use newly generated pairs to generate templates creating the template repository that allows for a wider variety of questions. For example, given a $< question - query >$ pair extracted from QALD-9th training dataset: "what is the population of Cairo?" and the corresponding query is:

SELECT ?answer

WHERE{*dbr*:*Cairo dbo*:*populationTotal ?answer.*}

Using the *transformer*, the following paraphrases are available for *LingTeQA*:

- "what is the population of Cairo?"
- "how many people are living in Cairo?"
- "how many people live now in Cairo?"
- "what is the population of Cairo, Egypt?"
- "What population does Cairo have?"
- "what is the population total of Cairo?"
- "what's the population of Cairo?"
- "What are the demographics of Cairo?"

Consequently, *LingTeQA* uses eight pairs of $< paraphrazedquestion, query >$ to generate eight templates instead of just one. With the extended template repository, our system can perform slightly better. Our experiment on QALD

[3]https://github.com/ramsrigouthamg/Paraphrase-any-question-with-T5-Text-To-Text-Transfer-Transformer

Table 3.3: LingTeQA results: QALD-9 DBpedia & QALD-7 Wikidata with and without paraphrases

Paraphrased	Dataset	macroP	macroR	macroF1
Yes	DBpedia	0.556	0.665	0.545
	Wikidata	0.646	0.739	0.646
No	DBpedia	0.526	0.642	0.535
	Wikidata	0.634	0.735	0.642

datasets shows some improvement. In particular, without using paraphrasing, the system can generate only 107 templates from provided 408 pairs of <*question-SPARQL query*> (see Section 3.5.2) whereas it can generate an extended template repository containing 631 templates by using paraphrased questions. *LingTeQA* uses the extended template repository to answer more questions in 9th QALD DBpedia and 7th-QALD Wikidata (88/150 questions over DBpedia, 72/100 questions over Wikidata), and with slightly higher precision, recall, and F1-measure as well. Table 3.3 gives the detailed results.

The paraphrasing process allows *LingTeQA* to better handle the variability found in natural language.

3.7 Question-Answering: Related work

In this section, we survey various approaches towards developing question answering systems. We will provide some main ideas underlying each of them and list some systems that has been adopting the approach.

3.7.1 Information Retrieval-Based Question-Answering Systems

Originally, question answering had a strong focus on textual data sources to find answers, relying mostly on information retrieval techniques[98]. In this approach, a QAS may broadly have three stages, i.e., question analysis: parsing, question classification and query reformulation for finding relevant text (documents, articles, web pages,...); document analysis: extract candidate

documents, identify answers; and answer analysis: extract candidate answers and rank the best one [30], [66].

Question classification: some QASs using statistical techniques such as support vector machine (SVM) classifiers, Bayesian classifiers, Maximum entropy models for predicting users' expected answer type. These models are trained on a corpus of questions or documents that has been annotated with the particular mentioned categories in the system. IBM's statistical QA system [49] utilized a statistical algorithm (maximum entropy model) for predicting the class of the answer desired by a question and the class of segments of text based on various N-gram or bag of words features while systems developed by Moschitti [77], Zhang and Zhao [121] had used SVM classifiers for question and answer categorization. Some of the knowledge based QA systems relied on the rule based mechanism. After applying general purpose NLP techniques, rules are further built to identify question classification features. Quarc developed by Riloff and Thelen [87] and Cqarc developed by Xiaoyan et al. [107] used heuristic rules that look for lexical and semantic clues in question to identify the question class.

Methods used for answer finding task vary from system to system. The IBM's statistical QAS used a two-pass approach. In the first pass, the system searches an encyclopedia database. The highest scoring passages were then used to create expanded queries by applying local context analysis technique. The expanded queries are applied in the second pass scoring of the TREC documents. Top sentences are then ranked by a maximum entropy based answer selection model. A QAS by Moschitti [77] implemented similarity measurement model that accounted on different features such as keyword similarity, length similarity, order similarity and distance similarity

3.7.2 Template-Based Question-Answering Systems

Template-based approach has been applied for structured data such as Database and Knowledge Bases (aka Knowledge Graphs). Query languages such as SQL and SPARQL[4] are standard ways of accessing the structure data. Therefore,

[4]https://www.w3.org/TR/rdf-sparql-query/

the main idea of the approach is the translation from users' questions into templates in form of $\lambda - calculus$, triple-based, or SPARQL templates. Then templates are converted into SQL/SPARQL queries. A benefit of templates is that the mappings to the KG are traceable and can be leveraged to generate explanations for the user to understand why she receives specific answers.

Template representation

$\lambda - calculus$ is a formal system in mathematical logic for expressing computation based on function abstraction and application using variable binding and substitution. It is used by Platypus [92]. For instance, Platypus represents question "Where was the inventor of dynamite born?" as
$\{y|\exists x < dynamite, inventor, x > \wedge < x, birthPlace, y >\}$

Triple-based are used in systems such as AquaLog [69], Power-Aqua [70]. For example, AquaLog translates the question *'Who is a Professor at the Knowledge Media Institute?'* into an intermediate triple
$< who,\ Professor,\ Knowledge\ Media\ Institute >$ then map the intermediate triple into ontology-compliant logical query

$< typeOf\ ?x\ Professor\_in\_Academia > \&\ < work\_in\_unit\ ?x\ KMi >$.

Power-Aqua transforms the question "Give me actors starring in movies directed by Clint Eastwood" into $<$ *actors*, *starring*, *movies* $>$ and $<$ *actors/movies*, *directed*, *Clint Eastwood* $>$.

Other systems use predefined generic templates. For instance, Aqqu [6] uses three SPARQL query templates: $< e1, r1, t >$, $< e1, r1, m >< m, r2, t >$, and $< e1, r1, m >< m, r2, e2 >< m, r3, t >$ where e is an entity placeholder, r is a relation placeholder, m is an intermediate object and t is the output variable.

Many systems employ templates containing $<$ *question* $-$ *query* $>$ pairs, guiding the mapping of utterance constituents onto query components. For example, PARALEX [31], a single-relation QA system, using question templates such as *'Who r the e books?'*, *'Who is the r of e?'* along with query templates in forms of r(?, e) or r(e, ?) where r is a relation and e is an entity. OQA [32] uses 10 hand-written templates, each of them consists of a question pattern

and a query pattern. The question pattern expressed using noun phrases (NP), auxiliary verbs (Aux), and ReVerb patterns (RV) while the query pattern is expressed using triple-based representation. '$Who/What\ RV_{rel}\ NP'_{arg}$ and $(?x,\ rel,\ arg)$ is a template example. QUINT [1] builds a bank of templates where each template $t = (u_t; q_t; m_t)$ composed of an utterance template u_t, a query template q_t, and an alignment m_t between the two indicated by shared ent, pred, and type annotations. For example, based on a pair of question u = 'Which actress played character Amy Squirrel on Bad Teacher?' and answer $A_u = \{LucyPunch\}$ the system builds a template as

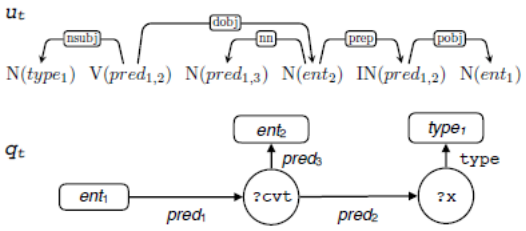


Figure 3.11: A template generated by QUINT

For an asked question, QALL-ME Framework [34] and AutoSPARQL TBSL [97] produce SPARQL-like templates that are SPARQL queries containing slots which are placeholders for KG's URIs. For example, AutoSPARQL TBSL [97] builds two SPARQL templates for the question *'How many films did Leonardo DiCaprio star in?'*. One of them is

$SELECT\ COUNT(?y)\ WHERE\{?x\ ?p\ ?y.\}$

with slots: $<?x,\ resource,\ Leonardo\ DiCaprio>$ and $<?p,\ property,\ films>$.

QALL-ME Framework [34] generates a question template *'Where can I see the movie [MOVIE]?'* paired with a SPARQL query template

$SELECT\ ?cinemaName$
$WHERE\{?movie\ qmo{:}name\ \text{“}[MOVIE]\text{”}.\ ?cinema\ qmo{:}showsMovie\ ?movie.$
$?cinema\ qmo{:}name\ ?cinemaName.\}$

for questions about the cinemas that show asked movies. Let us take a look how these templates can be generated.

Template generation:

Platypus [92] firstly parses a question with standard tools such as CoreNLP, SyntaxNet, and Spacy yielding a dependency tree. Next, it applies rules on the dependency tree to produce the logical representation of the question.

QUINT [1] and NEQA [2] uses a distant supervision model with training data are questions paired with their answer sets to learn templates with alignments between the constituents of the question utterance and the KG query.

AquaLog [69] firstly generates intermediate triples in form of term-relation and then classifies them based on syntactical annotations. Next, it produces an ontology-compliant logical query by using the structure of the ontology and information stored in the target KGs, as well as string similarity matching and lexical resources, such as WordNet and user's help for disambiguation.

Power-Aqua [70] firstly analyzes an asked question and translates it into its linguistic triple form. Next, it identifies ontologies that are likely to provide the information requested by the user. Then it matches the linguistic triple terms and lexically related words obtained from WordNet and the ontologies. Once the set of possible syntactic mappings has been identified, it checks its validity using a WordNet-based filtering methodology. After this process, it generates a set of Entity Mapping Tables where each table links a query term with a set of concepts mapped in the different domain ontologies.

AutoSPARQL TBSL [97] firstly obtains part-of-speech information provided by Stanford POS tagger on an asked question. It then looks up tokens from the question in a domain-independent lexical, adds to the lexical new entries if they do not exist. Next, it parses obtained lexical entries to construct semantic representations of the whole question. Finally, it translates the semantic representations into SPARQL templates.

Aqqu [6] uses conditional probability to map noun phrases in a question into Freebase entities. Based on entity matches it generates query candidates using the three predefined templates. For each the query template it selects relations from the query and match them with phrases in the question using

literal, derivation, synonym, and context matches. Finally, it uses learning-to-rank techniques to learn pair-wise comparison of query candidates to choose the best candidate query for answering the question.

Using templates:

PARALEX [31] and OQA [32] apply hand-written templates on paraphrased questions from an input question to produce query templates. They then populate the templates using a lexicon that encodes mappings from natural language to database concepts. Finally, they executes rewritten query against a KG to get final answer.

QUINT [1] performs light-weight template matching. The user utterance is considered matched with a template if a subgraph of its dependency parse tree is isomorphic to the question template considering their edge labels and the POS tags in their nodes. For each matching utterance template, QUINT [1] instantiates the corresponding query template based on alignment and the lexical L. Next it applies a learning-to-rank approach to rank obtained queries, and returns the highest ranking query as the one intended by the question.

Platypus [92] first finds the template that best matches a question using a classifier that ranks the logical representations (templates) according to their likelihood of being the correct interpretation of the question. Finally, it converts the representations into SPARQL queries, and executes one after the other on Wikidata, until one of them yields an answer.

3.7.3 Artificial Neural Networks-Based Systems

A widely used methodology of Artificial neural networks (ANN)-based QA systems is that they use an RNN to produce vector representations for a given question and candidate KG's subject entities and predicates associated with the question. The question is often encoded as the final state of an RNN whose input is a concatenation of vector representations of its words at a different level. Next, a probabilistic formulation or a similarity measurement is used to find the most matched subject-predicate pair for the question. Finally, the object of the triple whose subject and predicate are found in the previous step

is returned as the answer to the question.

The differences between QA systems lie in the techniques they adopted for finding candidate KG's subject entities and predicates. The system developed by Dai et al. [16] uses a conditional probability for inferring the implied relation from the given question first, then inferring mentioned subject given the inferred relation. These conditional probabilities are calculated based on the exponential of the dot product of vector representations of the condition (question/relation) and the event (relation/subject). The QA system introduced by Lukovnikov et al. [71] finds candidate subjects and predicates by first collecting the lowercased English entity labels from a target KG. All word n-grams of size 1 to L contained in question q are retrieved, filtered, and used for searching matching entities using rules. Matching entities are then ranked by the number of triples in the KG. Only m entities with the highest rank are added to the candidate set. Next, a candidate set of predicates is generated from triples whose subject belongs to the candidate subject entities. The QA system for answering simple questions built by Hao et al. [42] uses a BiLSTM-CRF model in the first stage to split a question utterance into mention span and question pattern. After revising obtained question pattern, each mention span is used to retrieve subject entities in KG whose names contain overlap words with the mentioned span. Then the system ranks the candidate subject entities according to each entity's longest consecutive common subsequence, top-M ranked entities are kept for each question. Afterward, each of the selected subject entities is used to collect associated predicates. In the next stage, given a pair of (mention span, question pattern) and sets of candidate subject entities and associated predicates retrieved in the previous stage, the system uses cosine similarity score between the encodings of string surface-form of the mentioned span and entities' names are calculated to rank candidate subject entities. The same similarity measure is used to rank the candidate predicates.

Different from the aforementioned ANN-based systems, the QA system proposed by Iyer et al. [50] used a neural sequence-to-sequence model to directly generate SQL queries from natural language questions. The model is

an encoder-decoder model with global attention where the anonymized utterance is encoded using a bidirectional LSTM network, then decoded to directly predict SQL query tokens. Fixed pre-trained word embeddings from word2vec are concatenated to the embeddings that are learned for source tokens from the training data. The decoder predicts a conditional probability distribution over possible values for the next SQL token given the previous tokens using a combination of the previous SQL token embedding, attention over the hidden states of the encoder network, and an attention signal from the previous time step. The preliminary semantic parser is initially trained, then iteratively improves this parser using user feedback and selective query annotation.

The ANN-based approach is easier to retrained or reused for a different domain, avoid error propagation but is hard to control because the systems are given the freedom to make a decision; thus, they may only be suitable for answering factoid questions rather than complex questions or questions with aggregations because they do not handle deeper linguistic phenomena such as quantification, negation, and superlatives, etc.

3.7.4 Graph-Based Question Answering Systems

Graph-based QA systems map semantic phrases in a question into KG items to construct the Directed Acyclic Graph (DAG). The systems then reduce the question answering problem to a subgraph matching problem.

Zou et al. [126] proposed a two-stage graph data-driven solution to answer a natural language question. In the question understanding stage, they interpret a natural language question N as a semantic query graph Q^S in which each edge denotes a semantic relation extracted from N. The semantic relation is a triple $< rel, arg1, arg2 >$, where rel is a relation phrase and its associated arguments are arg1 and arg2. In the query evaluation stage, they find subgraph matches of Q^S over RDF graph G based on the semantic similarity of the matching vertices and edge in Q^S and the subgraph match in G. Top-k subgraph matches with the largest scores computed from the confidence probabilities of each edge and vertex mapping are used for answering the question N.

Zhu et al. [124] proposed a three-stage graph traversal-based method for

answering non-aggregation questions. In the question understanding stage, they use an entity linking method to detect the mention-entity pairs, of which the mention is used for the phrase boundary identification. Next, they build a list of topological patterns to discover the structure by taking advantage of the parsing result of the query. In the Graph Traversal stage, they build a subgraph of the underlying knowledge graph rooted from entities found in the last stage. Then they use a jointly ranking method to find the most appropriate traversal path in the subgraph. The topological structure is used for semantic item mapping and judging traversal stop conditions. In the last stage, namely Focus Constraint, they extract a phrase describing the answer directly from the query, which is called a focus to help modify final path ranking scores. After the above three stages, the overall path candidates ranking list is obtained. The answers found along the path with the highest score will be returned.

Hu et al. [47] also introduced a graph data-driven approach for developing RDF question answering systems. They first apply Stanford Parser to a natural language question to obtain a dependency tree. They then extract semantic relations based on the dependency tree to build a semantic query graph by mapping the relation mentions and node phrases to candidate predicates/predicate paths and entities/classes, respectively. Next, they find top-k RDF subgraphs of the chosen RDF graph that matches the previously created semantic query graph with the highest matching scores.

Interactive Question Answering Systems

The flexibility, complexity, and ambiguity of natural language questions result in errors in translating natural language phrases into semantic items. Consequently, the errors degrade the performance of QA systems. A natural way of clarifying ambiguities is by asking some questions back to the users or let them interact with KGs allowing for manual mapping between natural language expressions and semantic items. Here are some interactive question answering systems:

Aqualog [69] is a QA system that asks users for helping disambiguation when analyzing user's queries. In particular, the user is asked to help choose

the right relation or instance.

FREyA [17], a Natural Language Interface for querying ontologies, aims at improving recall by enriching the domain lexicon from the user's vocabulary and improving precision by resolving ambiguities more effectively through the dialog. There are two kinds of dialog in FREyA. The disambiguation dialog involves the user resolving identified ambiguities. The mapping dialog involves the user to map a *Potential Ontology Concepts* (question terms/phrases) to the one of the suggested *Ontology Concepts* (KG instances/individuals, classes, properties).

SPARKLIS [35], a Semantic Web tool, helps users explore and query SPARQL endpoints by guiding them in the interactive building of questions and answers. It integrates Faceted Search (FS), Query Builders(QB), and Natural Language Interfaces (NLI). The NLI allows users to form questions step by step by selecting words that are generated via auto-completion manner. The FS and QB give a set of suggestions to refine the current selection, and users only have to pick a suggestion according to their preferences. The QB lists eligible constructs at each step enabling query completion with minimal syntax errors. However, to use the tool, users need to learn how to use it, but that after a short training, they can answer complex queries.

A system developed by Zheng et al. [123] lets users verify the ambiguities during query understanding to answer natural language questions over Knowledge Graphs. Given a question, their system firstly enumerates all possible candidate phrases and then finds their corresponding candidate mappings in the Knowledge Graph. The phrase mappings are assembled to form complex query structures based on the input question and the underlying knowledge graph. If the system does not understand the question for some ambiguities in the whole process (generating candidate phrase mappings and query structures), it will resort to the user by presenting her with the ambiguous candidates and letting her make choice.

IMPROVE-QA [122] aims for a better understanding of natural language questions and more precise-answer returning via users' feedback to the system. In particular, it learns from the translation from natural language questions

to queries and the users' feedback to generate more precise answers and avoid translation mistakes on answering subsequent questions using paraphrasing dictionaries.

NEQA [2], a template-based system, harnesses non-expert user feedback on an answer set generated as a response to a given question to expand its template repository.

IQA [120] incorporates user feedback in the question-answering process for semantic QA pipelines. First, IQA displays the current question along with a top-ranked query is provided in its natural language representation and a SPARQL representation. Using this part of the interface, the user can accept the top-ranked query. User issues the question Q first. Then IQA generates the question interpretation space and generated interaction options. The user is then simultaneously presented with the interaction option to accept/reject until a condition is met, for example, the user accepts the complete question interpretation, the question interpretation space is empty. The interaction option is expressed as an inquiry along with a candidate answer. The user can select from "yes"/"no"/ "don't know" answers to accept or reject the interaction option displayed. According to the user feedback, the interaction option and the top-ranked query are updated.

Conversational Question Answering Systems

Question-Answering systems provide a user-friendly means to find information needs from open KGs. On many occasions, users' information needs are not always phrased in well-formed and self-contained questions for one-shot processing. Users tend to ask follow-up questions that usually make complete sense only in conjunction with the conversation context: the previous question and the previous answer [65]. To better meet the users' information needs, QA systems need to maintain conversation context using entities and predicates seen so far and automatically inferring missing or ambiguous pieces for follow-up questions. Such QA systems are called conversational Question Answering ones.

Building conversational QA systems are challenging because of incomplete

follow-up questions, implicit entities or predicates, ungrammatical phrase, and implicit context from previous interactions [15].

To address the challenge, Kumar and Joshi proposed an approach of question completion aiming at creating syntactically correct full-fledged interrogative sentences from the user's inputs using retrieval-based sequence to sequence learning. They train their system using a labeled dataset containing a few thousand conversations, by decomposing the original problem into two simpler and independent problems. The first one focuses solely on selecting the candidate questions from a library of question templates (built offline using the small labeled conversations dataset). In the second one, they re-rank the selected candidate questions using a neural language model (trained on millions of unlabelled questions independently). By doing so their retrieval-based system can return a complete question for an incomplete follow-up question, given the conversation context: the previous question and the previous answer. [65].

Christmann et al. [15] developed a solution called CONVEX, an unsupervised method, that can answer incomplete questions over a Knowledge Graph. Their core method is a graph exploration algorithm that judiciously expands a frontier to find candidate answers for the initial question. In particular, the question is used to identify a small subgraph of the KG for retrieving answers. For incomplete and ungrammatical follow-up questions, they capture the context in the form of a subgraph as well, and they dynamically maintain it as the conversation proceeds. This way, relevant entities and predicates from previous turns are kept in the gradually expanding context.

Conversational Question Answering is a challenging but promising task. We anticipate that it will become one of the active research trends in the future.

Hybrid Question Answering Systems

HAWK [100] is a hybrid QA system that uses predicate-argument representations of questions to derive equivalent combinations of SPARQL query fragments and text queries. The system integrates the results of the text queries into SPARQL to generate a formal interpretation of the query. HAWK im-

plements an 8-step pipeline in which the first four steps create a predicate-argument graph annotated with resources from the Linked Data Web. The next two steps assign semantic meaning to nodes and generate basic triple patterns for each component of the input query accordingly to a multitude of features creating a set of SPARQL queries containing text operators as well as triple patterns. The last two steps discard queries using several rules and rank queries sing extensible feature vectors and cosine similarity.

A QA system introduced by Xu et al. [108] is another hybrid system that exploits both structured data from DBpedia and Freebase and free text from Wikipedia. The system firstly performs entity linking and relation extraction then uses an integer linear program (ILP) model to solve the disambiguation among entities and relations across text and KGs. The entity linking is done by using DBpedia Lookup and S-MART to retrieve top 10 entities from DBpedia and Freebase, respectively as candidate entities while relation extraction is implemented using a MultiChannel Convolutional Neural Networks (MCCNNs) model to map relational phrases to KG predicates and paraphrase model to predict textual relations from the relational phrases.

Platypus [92] is also a hybrid QA system that combines a set of transformation rules based on sentence grammars and query templates and slot filling. The system works in three steps. In the first step, its analyzer converts the natural language question into one or several internal logical representations using rules. In the second step, its template analyzer ranks logical representations (templates) according to their likelihood of being the correct interpretation of the question. In the last step, the representations are converted into SPARQL and executed one after the other on Wikidata, until one of them yields an answer.

Multilingual Question Answering Systems

Since both publishing data in languages other than English as well as users who access this data and speak native languages other than English are growing substantially, multilingual question-answering systems have been gaining a great deal of attention from the Semantic Web community. The systems

provided below are very recently proposed ones.

Platypus [92] is a Multilingual Question Answering Platform for Wikidata. It can answer questions posed in English or French.

WDAqua-core0 [22] is a rule-based system that supports both full natural language queries as well as keyword queries posed in English on DBpedia and English, French, German, and Italian on Wikidata.

MuG-QA [125] is a Multilingual Grammatical Question Answering system that can answer questions posed in English, German, Italian and French over DBpedia. Their natural language modeling and parsing are implemented using Grammatical Framework in which concrete syntaxes for the German, Italian and French languages, operating with the same categories and concepts of the common abstract syntax. Once a natural language question is parsed, the resulting abstract grammar tree is matched with the knowledge base schema and contents to formulate a SPARQL query.

LAMA [79] is a multilingual (English and French) QA system that uses a set of lexico-syntactic patterns used to generate the SPARQL queries. LAMA's pipeline is composed of 3 main steps: Pre-Processing, Syntax Tree Representation, and SPARQL Query Generation.

WDAqua-core1 [19] is a multilingual and KG-agnostic QA system. It is able to query several knowledge bases simultaneously, in different languages. According to the authors by collecting lexicalizations for different languages and KGs and using Apache Lucene, their system can easily handle multilingualism.

3.8 Conclusion

We have introduced many developed question-answering systems with various approaches to show the diversity in question answering. Although we have adopted the template-based method to develop LingTeQA, our system is different from other template-based systems.

An integral part of the system is a template repository that contains pairs $<$*question template*$-$*SPARQL query template*$>$. A newly asked question is

converted into a question template and checked against entries of the repository. As a result, a corresponding SPARQL query template is retrieved. This template is used to generate a SPARQL query to answer the question.

To automatically build the repository and perform all tasks leading to generate and execute a SPARQL query, we have developed algorithms for 1) interpreting natural language questions, so questions with the same syntactical structure are mapped into the same pair of templates; 2) generalizing SPARQL queries; and 3) constructing question and query templates.

The constructed queries are traceable and interpretable. They can be leveraged to generate explanations for users so they understand why specific answers were obtained.

Examining the LingTeQA's performance on generating templates from question-query pairs in QALD's training dataset and answering questions in QALD's test dataset, we noticed that mapping question's constituents into KG's properties are the most challenging task. Although KG's properties usually bear mnemonic names, their only actual connection to natural language is by the labels that are attached to them. These labels often provide a canonical way to refer to the URI, but usually do not account for lexical variation. Although the vocabulary of natural language and the vocabulary used by the data overlap, the expressions a user uses often differ from the labels attached to the data [98]. As a result, how well the system answers a question heavily depends upon its syntactic structure and phrases used to express its meaning.

Despite being sensitive to the wording of questions, with generated templates, our system can answer a variety of questions, including questions containing imprecise concepts (linguistic terms). In addition, it can generate linguistic summaries to answer questions whose answers are long lists of numbers. We present the processes of answering such questions in subsequent chapters.

Chapter 4

Human-centric Question Answering System with iPad-based Interface for Defining Linguistic Terms

Searching for information and discovering useful from a large amount of stored data become an everyday activity of many users. On many occasions, these are tedious tasks requiring knowledge related to interacting with data repositories and knowing how to define linguistic terms representing personal perceptions of concepts understood by users. The users want to use such concepts to learn how well, i.e., to what degree, the analyzed data satisfies these concepts. In other words, the users want to know how well the analyzed data matches their perceptions of concepts.

However, human-computer interaction is crude and far from natural. Users are forced to interact with data repositories using languages understood by machines. A simple way to learn more about phenomena represented by data can be done via representing the data as human-perceived concepts and enabling a human-like interpretation of it. The users should be able to use linguistic terms –for example LARGE, SLOW, MOST –as their representations of concepts in order to gain a better understanding of data. Yet, other issues arise: how to enter definitions of such terms, how to incorporate individual's understanding of their meanings, how to ensure their proper interpretation, and of course, how to do all this in an easy and simple way.

In this chapter, we present and describe an iPad-based software that enables an easy procedure of defining linguistic terms. The application –called Tablet input of Fuzzy Sets (TiFS) – allows users to define terms in a simple way via drawing their 'shapes' using fingers. We provide a detailed application of Tablet input of Fuzzy Sets (TiFS) in question-answering system.

4.1 Linguistic Terms and Quantifiers as Fuzzy Sets defined with iPad

An iPad-based application described here, TiFS [113], addresses a need for a system simplifying the process of defining fuzzy sets. It allows users to define, or shall we say to draw, shapes of membership functions using their fingers. The simplicity of the graphical interface of TiFS , and the need for entering only basic information makes the process of constructing fuzzy sets very convenient and straightforward.

4.1.1 Defining Linguistic Terms using TiFS

A graphical interface of TiFS is shown in Figure 4.1. It allows for defining linguistic terms. It is a simple interface that requires a minimum amount of input data. A user starts with providing a name of the domain on which a given linguistic term should be defined –the input field DomainName. Due to the fact that multiple terms can be defined on the domain, the user has to enter a name of a term she intends to define –the input field Summarizer/Linguistic Term . Also, the user has to provide information about the domain range –the input fields Range From: and To: –that allows the user to enter the minimum and maximum values of the domain.

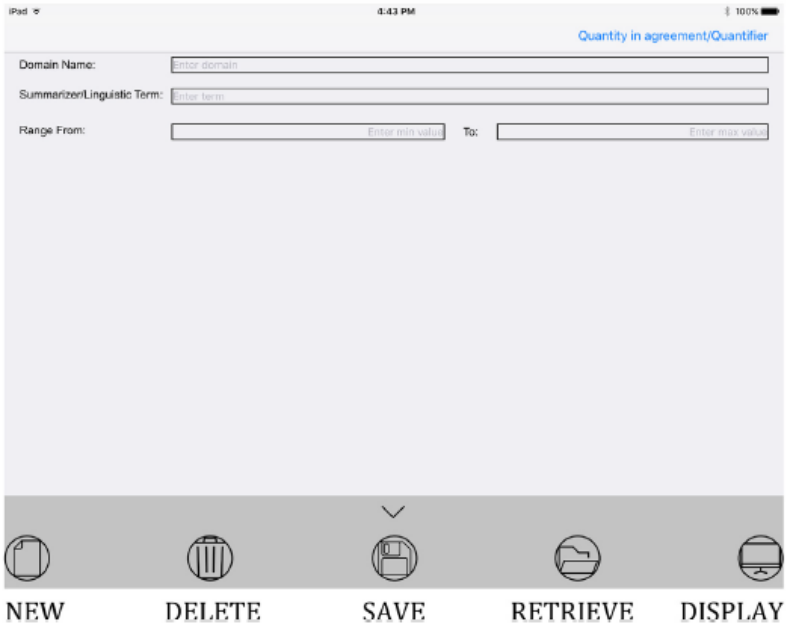


Figure 4.1: Interface of iPad application TiFS for defining linguistic terms.

In order to manage previously defined terms on multiple domains, the TiFS provides the user with a number of options located at the bottom of the screen. There are (from left to right, Figure 4.1): NEW –to clean the content of input fields and prepare the application for new definitions of terms; DELETE –to remove an already existing definition of the term that is identified by the domain's name and the term's name; SAVE –to store the entered definition of a term into the application storage, the stored data includes: a domain name, a term name, range and values of the term, and its shape entered by a user; RETRIEVE –to load the previously defined term from storage based on the domain and term names, and show the membership function shape on the screen; DISPLAY –to display fuzzy function membership values of the previously defined and stored term, the term to be displayed is identified by the domain and term names.

4.1.2 Web Interface for Defining Linguistic Terms

We have implemented *LingTeQA* [95] as a Web application. It is accessible at ***www.lingteqa.site***, and it contains a user-interface, based on *TiFS*, for entering membership functions defining linguistic terms.

Whenever the system answers a question containing a linguistic term, it collects data from a specified KG by executing a constructed query. It then prepares a coordinate plane for a user to enter their understanding of the term. To make the x-axis relevant, the system determines a proper range: it finds out the minimum and the maximum of obtained results, and uses them to scale the x-axis properly. A screenshot of the interface when answering the question "give me large cities in Poland by population" with data collected from DBpedia is shown in Figure 4.2

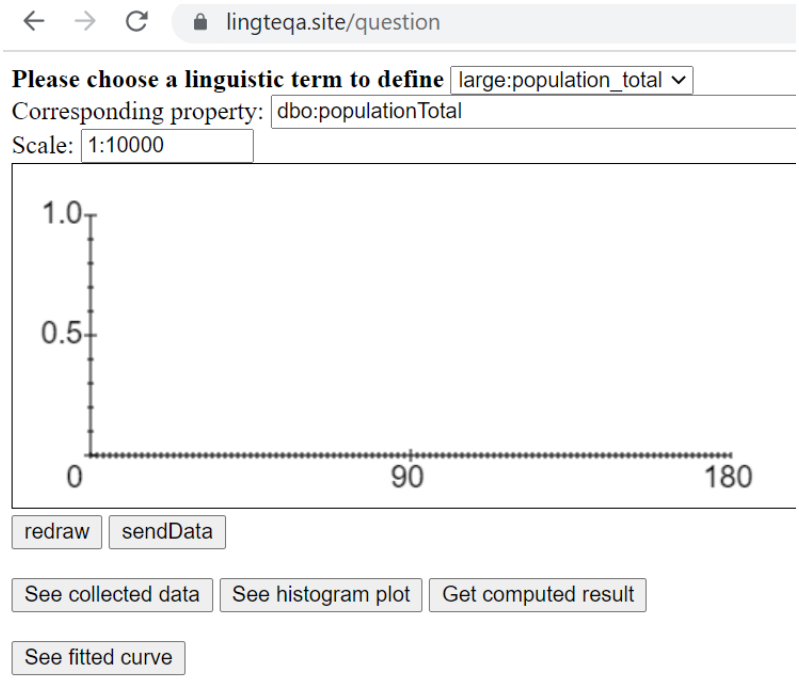


Figure 4.2: TiFS-based Web Interface for Defining Linguistic Terms

The user can click on **See collected data** button to see actual data or **See histogram plot** one to see the distribution of data. Screenshots of actually collected data and its histogram plot are given in Figure 4.3 and Figure 4.4, respectively.

	city	population_total
76	http://dbpedia.org/resource/Warsaw	1793579.0
48	http://dbpedia.org/resource/Łódź	679941.0
9	http://dbpedia.org/resource/Wrocław	643782.0
68	http://dbpedia.org/resource/Poznań	534813.0
13	http://dbpedia.org/resource/Gdańsk	470907.0

Figure 4.3: Five Polish cities with largest population

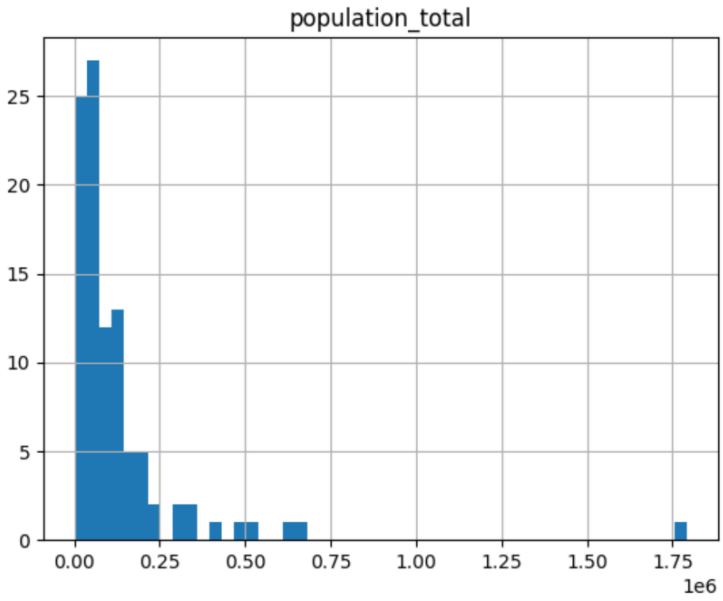


Figure 4.4: Histogram plot of Polish cities' population

The user draws or redraws (by clicking on **redraw** button) a shape of membership function on the provided coordinate plane until being satisfied with the shape representing the term. He/she submits the drawn membership function by clicking on the **send Data** button.

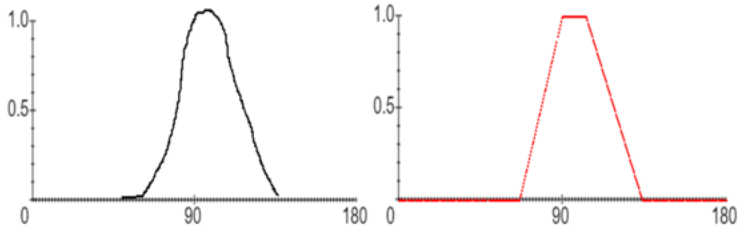
4.1.3 Function Fitting

Although *TiFS* provides users with an easy-to-use interface to draw 'shapes' of membership functions, their drawings will never be perfect. Therefore, function fitting (aka curve fitting) is a needed technique to adjust values of the function parameters to best describe a set of data determined by the user-drawn shape.

There are commonly used categories of membership functions. They are triangular, trapezoidal, Γ-shape, S-shape, Gaussian, and Exponential-like membership functions. All of them are defined in the universe of real numbers. In our system, we adopt a parametric fitting approach due to the fact that forms for the functions together with their parameters are known in advance.

We use *Scipy*, a large Python library of functions used for scientific analysis, for the curve fitting purpose. In particular, we use the function *curve_fit*. This function returns two items – they are the best-fit parameters of a fixed-form function. We also use *r2_score* function in *sklearn* (another Python library) to select a best fitted function to the data provided by the user.

A user-drawn membership function and corresponding function obtained after the fitting process are given in Figure 4.5



(a) (b)

Figure 4.5: user-drawn membership function (a); and system-fitting one (b).

Hereafter, when we use membership function defined by users, we mean and use their system-fitted versions.

4.2 Questions with Linguistic Terms

As a human, we can easily detect, capture meanings of linguistic concepts. However, how a machine can determine *velocity* as the descriptor of *high speed* while *height* or *tallness* as the descriptor of *high building*. This section will highlight the procedure that allows our QA system: 1) detect linguistic terms in a given question; 2) identify a descriptor (a KG property) for a detected linguistic term.

Algorithm 1 Identify and map linguistic terms to KG's properties

procedure DETECT LINGUISTIC TERMS(Q : $auserquestion; wn$: $WordNet; KG : aknowledgegraph$)
  $POSofQ \leftarrow POS\_tagger(Q)$
  $adjPhrases \leftarrow ExtractAdjPhrase(POSofQ)$
  $linguisticTerms \leftarrow \emptyset$
  while $i <= len(adjPhrases)$ **do**
    $adj, noun \leftarrow splitAdjPhrase(adjPhrases[i])$
    $attributes \leftarrow getAttributes(adj, wn)$
    if $attributes \neq \emptyset$ **then**
      $pro \leftarrow noun2pro(noun, KG)$
      if $pro \neq \emptyset$ **then**
        $dic.add(adjPhrases[i], pro)$
      else
        $cla \leftarrow noun2class(noun, KG)$
        if $cla \neq \emptyset$ **then**
          $instances \leftarrow getInstances(cla, KG)$
          $superlative \leftarrow getSuperlative(adj, wn)$
          $nounsInKG \leftarrow getMostNoun(instances, KG, noun, superlative)$
          $pro \leftarrow noun2pro(nounsInKG, KG)$
          if $pro \neq \emptyset$ **then**
            for $(p \in pro)$ **do**
              $dic.add(adjPhrases[i], p)$
            end for
          end if
        end if
      end if
    end if
    $i \leftarrow i + 1$
  end while
  $return\ dic$
end procedure

4.3 Defining Fuzzy Sets for QA System

Linguistic concepts are both vague and context-dependent. The meanings of *old* change when applied to different objects (e.g., a car, a house, a person); the concept *low income* may have different meanings when applied to Canadian and Vietnamese.

For the aforementioned reasons, we choose a direct user-driven approach that not only addresses the issues but also elicits knowledge from experts. However, answering questions in the user-driven methods that will be presented in more detail in Section 4.5.1 becomes a tedious job if there is a big number of elements of the universe of discourse because the asked expert(s) has to answer too many questions. They might also accidentally eliminate the gradual transition between membership grades of adjacent elements of the universe of discourse that is the desired characteristic of fuzzy sets because locally and arbitrarily value(s) is assigned to each element or pair of elements of the universe of discourse.

In line with user-driven approaches, we provide a Web-based application that provides a convenient and user-friendly interface for entering – using fingers – definitions of linguistic terms and quantifiers required for linguistic-based analysis of data. To be more precise, firstly the expert can globally assign degrees of membership for all elements once at a time by drawing a shape that is the plot of the corresponding membership function. Second, the continuity of the shape guarantees a smooth transition between membership grades for adjacent elements.

Due to the imperfection of human drawing, the drawn shape is then fitted to seven most commonly used categories of membership functions (triangular, trapezoidal, Γ-shaped, S-shaped, bell-shaped, exponential-like, and polynomial). A membership function along with its parameters is computed using a least-square curve fitting algorithm.

The method of Web-based construction of fuzzy set representing a linguistic term includes 5 steps.

Step 1. Identify a universe of discourse over which a variable that is the

descriptor of the concept is defined.

Step 2. Collect available values for the variable.

Step 3. A histogram plot is introduced to an expert who is expected to assign membership grades for elements of the universe of discourse so that he will see the distribution of the variable, the range of values.

Step 4. The expert will be introduced to a framework that allows him or her to draw a shape representing his or her proposal membership function that captures the meaning of the linguistic term. The shape will result in a set of pairs $(x, A(x))$.

Step 5. The set of pairs $(x, A(x))$ is fed to a least-square curve fitting algorithm that fits the data to seven predefined classes of membership functions.

4.4 Answering Questions with Linguistic Terms

4.4.1 Method

A user's question containing a linguistic term requires some extra processing steps in its answering process than one without linguistic terms. We explain the process of answering such a question in a diagram given in Figure 4.6

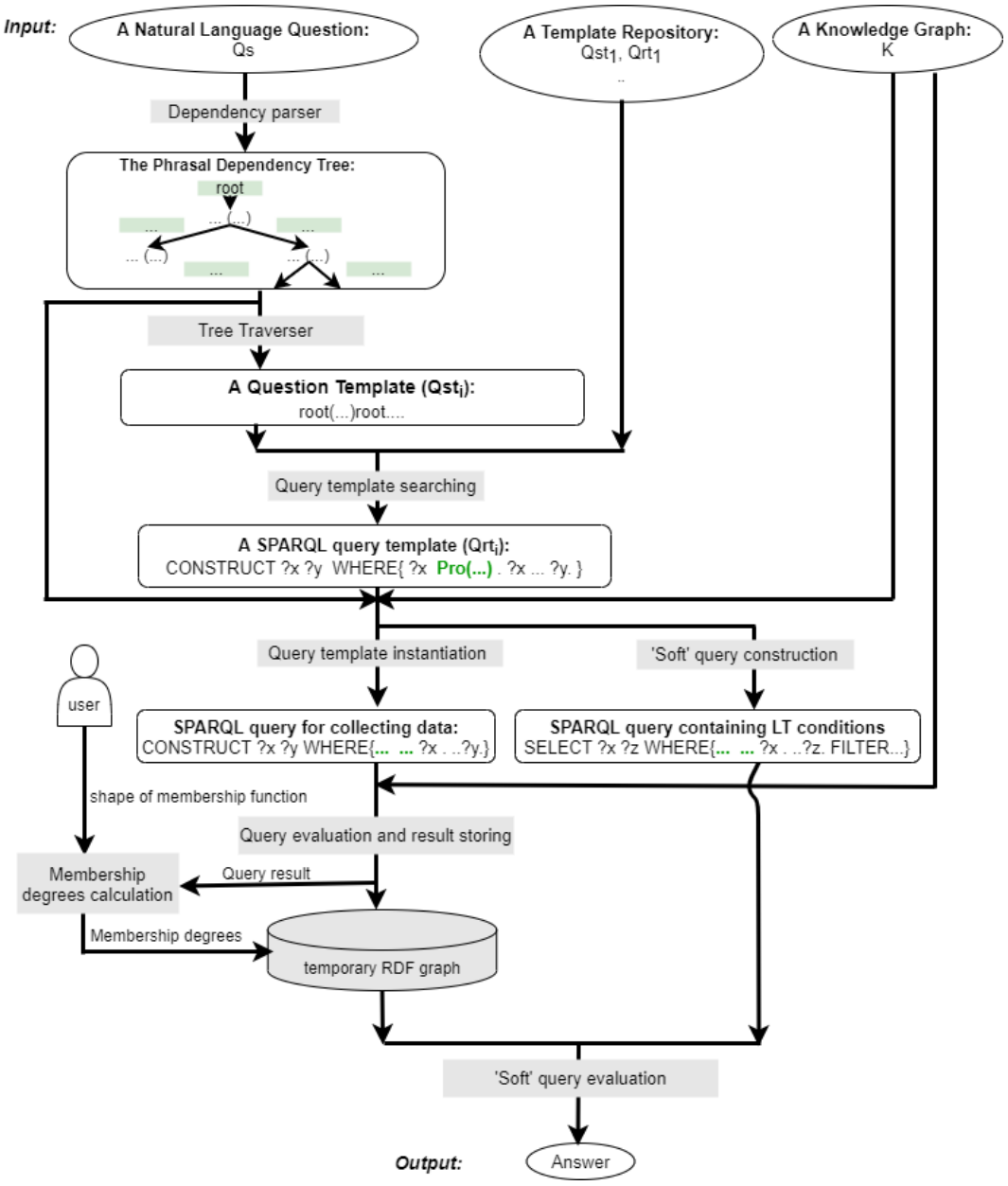


Figure 4.6: LingTeQA's process of answering a question containing linguistic term

4.4.2 Example and Analysis

We illustrate the process of answering a user question that contains a linguistic term by taking the question "give me large cities in Canada by population" as an example.

The detected linguistic term in the question by LingTeQA is *'large'*. The

phrasal dependency tree of the question is given in Figure 4.7.

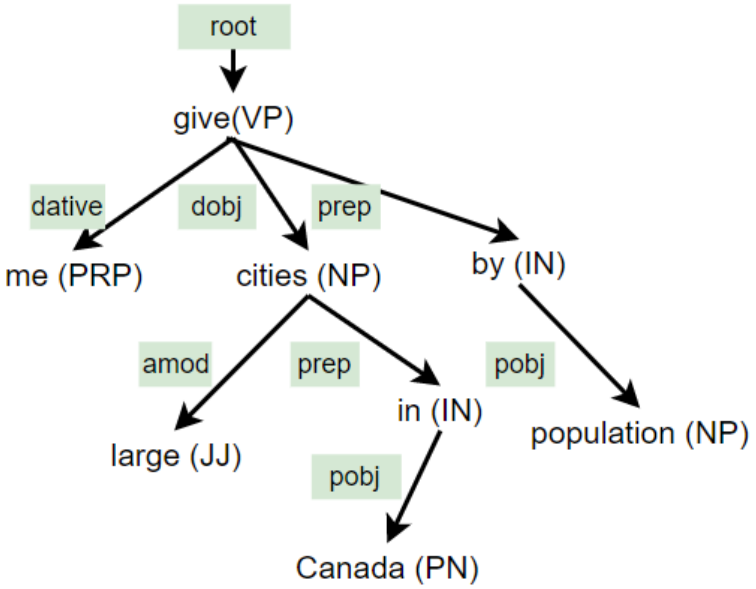


Figure 4.7: Phrasal dependency tree of a illustrative question containing a linguistic term

The question template (broken into two lines due to the lack of space) of the question is:

$root(VP) root.dative(PRP) root.dobj(NP) root.prep(IN) root.dobj.amod(JJ)$
$root.dobj.prep(IN) root.prep.pobj(NP) root.dobj.prep.pobj(PN)$

LingTeQA found a corresponding SPARQL query template by searching the question template in its template repository. The query template is:

$SELECT\ DISTINCT\ ?a\ ?b$
$WHERE\{?a\ Pro(type)\ Cla(root.dobj).\ ?a\ Pro(root.prep.pobj)\ ?b.$
$?a\ Pro(root.dobj.prep)\ Res(root.dobj.prep.pobj).\}$

LingTeQA then instantiated the found query template resulting in a SPARQL query for collecting data (cities and their population) from DBpedia. The constructed query (Qr1) is

$SELECT\ DISTINCT\ ?city\ ?populationTotal$
$WHERE\{?city\ rdf{:}type\ dbo{:}City.\ ?city\ dbo{:}populationTotal\ ?populationTotal.$
$?city\ dbo{:}country\ dbr{:}Canada.\}$

It also constructed a 'soft' query (Qr2) containing linguistic filter condition, as following

$SELECT\ DISTINCT\ ?city\ (?populationTotal\ as\ ?MembershipGrade)$
$WHERE\{?a\ rdf{:}type\ dbo{:}City.\ ?a\ dbo{:}populationTotal\ ?populationTotal.$
$FILTER(LANG(?populationTotal) =' large').\}$

where prefixes used in the queries are defined as follows

PREFIX dbo: <https://dbpedia.org/ontology/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

By evaluating the query Qr1 against DBpedia, LingTeQA received a list of cities along with their population. First five cities in the query's result are given in Table 4.1

Table 4.1: Canadian cities and population as query's result evaluated against DBpedia

city	populationTotal
http://dbpedia.org/resource/Montreal	1704694
http://dbpedia.org/resource/Ottawa	934243
http://dbpedia.org/resource/Winnipeg	705244
http://dbpedia.org/resource/Vancouver	631486
http://dbpedia.org/resource/Quebec_City	531902

The collected data are stored in an RDF graph (G) whose graphical representation is given in Figure 4.8.

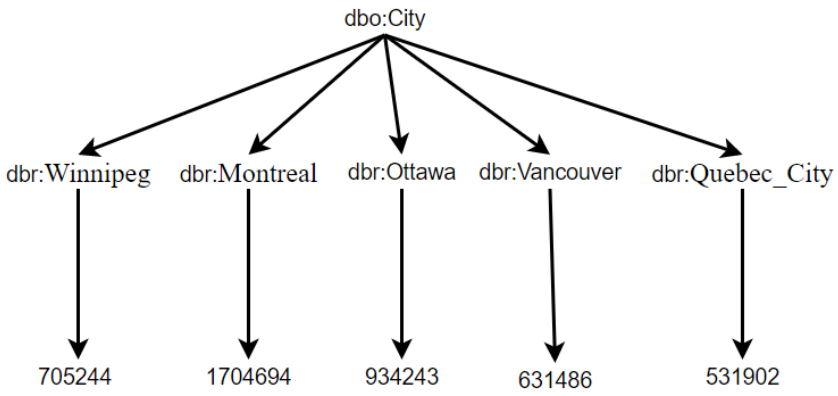


Figure 4.8: A temporary RDF graph storing data for answering linguistic-term-question

The system set a range in a coordinate based on the collected data for the user to draw a 'shape' of membership function describing the linguistic term 'large'. Figure 4.9 shows a user-drawn fitted 'shape' describing the term 'large'.

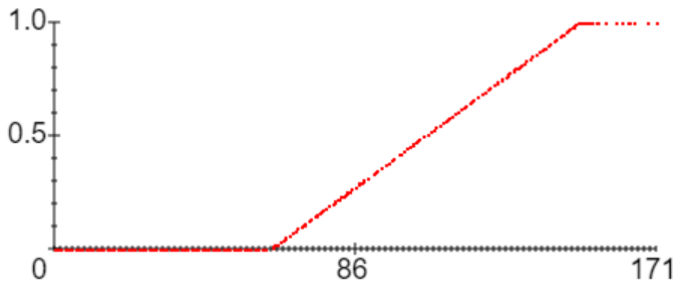


Figure 4.9: 'shape' of membership function drawn by user describing linguistic term 'large' (10k people)

Based on the fitted fuzzy set, LingTeQA calculated membership grades corresponding to the population of cities. The system then added triples whose objects are calculated values to resources corresponding to the collected cities in an RDF graph G. We visualize the newly updated RDF graph G as in Figure 4.10

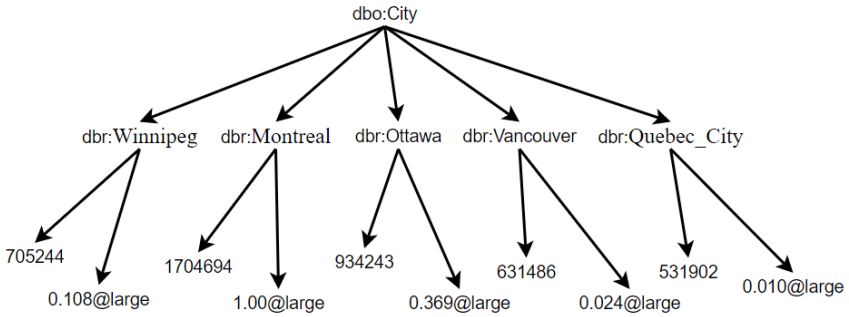


Figure 4.10: An RDF graph storing data for answering question containing linguistic term

By evaluating the constructed 'soft' query Qr2, LingTeQA provided a list of the top 5 cities whose membership grades are highest to the term 'large' based on its meaning provided by the user. A screenshot of the result is given in Figure 4.11

lingteqa.site/runsquery

	city	MembershipGrade
31	http://dbpedia.org/resource/Montreal	1.000
286	http://dbpedia.org/resource/Ottawa	0.369
45	http://dbpedia.org/resource/Winnipeg	0.108
171	http://dbpedia.org/resource/Vancouver	0.024
195	http://dbpedia.org/resource/Richmond,_British_Columbia	0.011

Figure 4.11: An answer to question "give me large cities in Canada by population based on user-provided membership function

However, if the user draws an alternative shape of the membership function as illustrated in Figure 4.12 then the user will receive a different result as shown in Figure 4.13

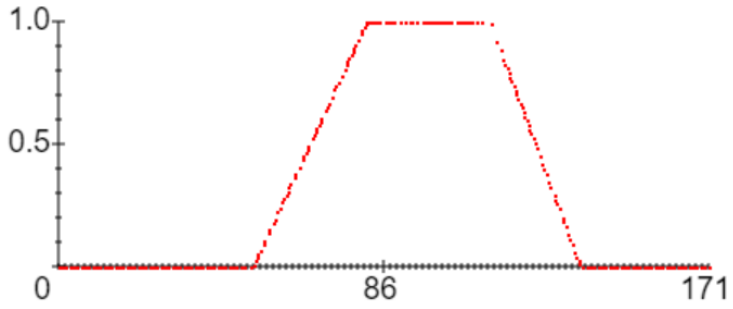


Figure 4.12: 'shape' of membership function drawn by user describing linguistic term 'large' (10k people)

lingteqa.site/runsquery

	city	MembershipGrade
286	http://dbpedia.org/resource/Ottawa	1.000
45	http://dbpedia.org/resource/Winnipeg	0.655
171	http://dbpedia.org/resource/Vancouver	0.409
241	http://dbpedia.org/resource/Quebec_City	0.076
0	http://dbpedia.org/resource/Barkmere,_Quebec	0.000

Figure 4.13: An answer to question "give me large cities in Canada by population based on an alternative user-provided membership function

4.5 Related work

An important aspect of utilization of fuzzy sets and fuzzy set-based technologies in multiple areas of industrial and commercial applications is related to their construction processes. Since the introduction of the concept of fuzzy sets, there are multiple examples of methods and techniques addressing an issue of building most suitable fuzzy sets, it means determining shapes of membership functions.

4.5.1 Construction of Fuzzy Sets

In general, there are several techniques for constructing fuzzy sets. They can be classified into user-driven and data-driven techniques.

The user-driven approaches can be further divided into direct methods and indirect methods.

In a direct method, an expert is expected to assign to each given element $x \in X$ a membership grade A(x) that, according to his or her opinion, best captures the meaning of the linguistic term represented by the fuzzy set A. This can be done by either defining the membership function completely in terms of a justifiable mathematical formula or exemplifying it for some selected elements of X by answering questions such as "what is the degree of membership of x in A" or " what is the degree of compatibility of x with L_A" [60].

In an indirect method, an expert is asked questions of the form 'what are elements of X which belong to fuzzy set A at degree not lower than α?' where α is a certain level (threshold) of membership grades in [0, 1]. The elements are identified by the expert form the corresponding α-cut of A. By repeating the process for several selected values of α a fuzzy set is finally constructed from the α-cuts. Another method in this group involves a group of experts. Each of them is expected to answer questions: 'does x belong to concept represented by a fuzzy set A?' for each element $x \in X$. The membership grade A(x) is then calculated as the ratio of number of 'yes' answers to number of asked experts. The simplicity of these methods is their advantages, however, they could exhibit a lack of continuity because the membership grades are separately computed for elements of the universal of discourse [84].

The priority method introduced by Saaty [88] forms another interesting alternative in which an expert is requested to compare elements in X in pairs according to their relative weights of belonging to A with a scale of 5, 7, or 9 levels. The expert will assign a high value of the available scale to the entry of a so-called reciprocal matrix at position of row ith and column jth if x_i is strongly preferred to x_j when being considered in context of the fuzzy set whose membership function we would like to estimate. The value of 1 at (i,j) position indicates that x_i and x_j are equally preferred. The eigenvector associated with the largest eigenvalue of the reciprocal matrix is then the needed fuzzy set. This method of constructing fuzzy set helps the expert focus

on only two elements once at a time thus reducing uncertainty and hesitation while leading to the higher level of consistency [84].

In the data-driven approaches, fuzzy sets can be formed on a basis of numeric data through their clustering. Fuzzy C-Means (FCM) is one of the commonly used mechanisms of fuzzy clustering. FCM clustering is completed through a sequence of iterations starting from a randomly initialized partition matrix and carry out the updating of clusters' prototypes and the partition matrix until a certain termination criterion has been satisfied. The final partition matrix indicates the way of allocation of the data to corresponding clusters. An entry u_{ik} is the membership degree of data x_k in the ith cluster[84]. There is also number of data-driven methods in literature. They differ in complexity of construction processes. Among variety of methods, membership functions are constructed using statistical and probability-based algorithms, different clustering algorithms, entropy, and evolutionary computation [43][101][14]. In particular, the authors of [101] describe an unsupervised technique that uses self-organizing maps to generate fuzzy membership function. Another unsupervised method is proposed in [14]. The authors propose method based on bandwidth mean-shift and robust statistics to construct membership functions. They use it to build triangular and trapezoidal functions representing underlying data. Also, the technique can determine a number of such functions. Chen et al. [14] proposes a gradient pre-shaped fuzzy C-means (GradPFCM) algorithm to generate better transparent membership functions. GradPFCM will preserve the predefined transparent shapes of membership functions during the process of the gradient descent based optimization of the clustering algorithm.

Fuzzy sets constructed by using aforementioned methods are almost invariably normalized, convex, and distinct. While those properties are undoubtedly useful in many application, they limit the selection of general shapes that may be used for representing membership functions for linguistic terms [39].

4.5.2 Fuzzy Queries and Relational Databases

To our best knowledge, *LingTeQA* is the first system that can answer questions containing linguistic terms over RDF knowledge graphs. However, fuzzy query languages extending the standard query language (SQL) and tools for fuzzy querying to relational databases were already proposed some time ago. Here we name a few noticeable ones.

FQUERY [52]–[54], [57], a family of fuzzy-logic-based querying systems, was developed and implemented by Kacprzyk and collaborators as a Microsoft Access "add-ons" to extend its capabilities of handling fuzzy terms. FQUERY uses a set of predefined fuzzy terms maintained and developed by users. Fuzzy values are defined as fuzzy sets on [-10, 10] interval, whereas the fuzzy linguistic quantifiers are defined as fuzzy sets on [0, 10] interval instead of the original unit one. The membership functions of fuzzy values are trapezoidal. The system has been successfully applied in querying databases and data summarization with linguistic terms.

Summary SQL [86] is a fuzzy query language introduced by Rasmussen and Yager. In Summary SQL, an attribute in a database (DB) is associated with a collection of fuzzy concepts defined via membership functions as fuzzy subsets over the attribute domain. Summary SQL allows for using linguistic terms in a query. For example, "select all persons where the height is tall" is such a fuzzy query. The query's result is a ranked fuzzy subset over the elements from the DB, including objects and their degree of satisfaction with the question. However, the authors did not mention how to enter membership functions defining fuzzy concepts.

Bosc and Pivert introduced a fuzzy set-based extension of the query language SQL called SQLf [11]. SQLf recognizes such extensions as selection, join, and projection with fuzzy conditions. They can also be applied to nesting operators and set-oriented operators. In addition, the language allows for the partitioning of relations involving groups based on fuzzy quantifiers.

Recently, Zadrożzny and Kacprzyk presented the basic structure of standard (crisp) analytic functions use in SQL syntax to elucidate the potential

of extending it using linguistic terms [119]. In particular, they started with a proposed method for defining "fuzzy" grouping and formulas for executing aggregate operators (COUNT, AVG, SUM) against resulting "fuzzy groups" of rows. The COUNT function is computed using a sigma count. The AVG function can be calculated using a weighted averaging or a Weighted Ordered Weighted Averaging Operators (WOWA). The SUM operator is calculated as a weighted sum derived from the AVG operator. After that, they introduced a new approach to the use of fuzzy terms in analytic functions creating flexible analytic clauses that enable computation of aggregated values in the context of a single row. The proposed extensions to the analytic functions have a potential application in generating more sophisticated linguistic summaries of data in relational databases.

4.6 Conclusion

The process of constructing SPARQL queries to extract information from RDF datasets is a challenging task. As far as we know, the existing state-of-the-art QA systems can answer simple questions, yet questions with imprecise linguistic terms are not considered.

We have introduced a system capable of answering questions with linguistic terms represented by user-defined membership functions. This system has been achieved by automatically performing such tasks as: 1) constructing a query to collect 'intermediate' data; 2) providing a user-friendly interface for users to draw membership functions enabling personalization of linguistic terms [113]; and 3) transforming obtained data into answers expected by a user.

The user-friendly interface allows users to draw shapes of membership functions associated with fuzzy terms representing terms and concepts. A simple process of inputting shapes as definitions of membership functions allows the users to quickly modify and/or change their definitions of concepts. We have illustrated the usefulness and easiness of using the interface to the question-answering process.

Chapter 5

Data Fusion

More and more Knowledge Graphs are available on Linked Open Data cloud. Many of them are cross-domain datasets describing millions of things. Although a single Knowledge Graph may contain billions of facts, it may not provide sufficient information for a question answering system.

An additional ability to span over multiple Knowledge Graphs (KGs) would further increase the usefulness and potentially comprehensiveness of the system's responses. Multiple different KGs, as much as they can complement each other regarding the lack of specific information, quite often contain inconsistent pieces of information. This makes data fusing a challenging task.

Although many question answering systems over KGs have been developing thus far, very few systems that collect and resolve conflicting multi-sources data according to surveys [45][20][44].

In this chapter, we propose a new approach to RDF data fusion applied on question answering: Veracity of collected data values from a Knowledge Graph is formulated based on their similarity with other collected values from other Knowledge Graphs and the trustworthiness of its sources–Knowledge Graphs. The truth value is then chosen based on calculated veracity. Next, the trustworthiness of the Knowledge Graphs is updated accordingly to the truth value. We illustrate the process of data fusion and its usefulness by a real example. We also provide a literature review on the data fusion task.

5.1 Fusion of Numerical Data

The process of fusing information/data presented in this chapter focuses on Resource Description Framework (RDF) data. In general, we propose an approach to identify the most reliable piece of information/data based on the data collected from multiple RDF Knowledge Graphs.

5.1.1 Motivation

Items described by triples in diferent KGs are highly overlapping. Although each of them is represented by a very different name (to be precise by a different Uniform Resource Identifier (URI)) they are explicitly linked by the property *owl:sameAs* – and this means they all represent the same items, see Figure 5.1 for illustration of that.

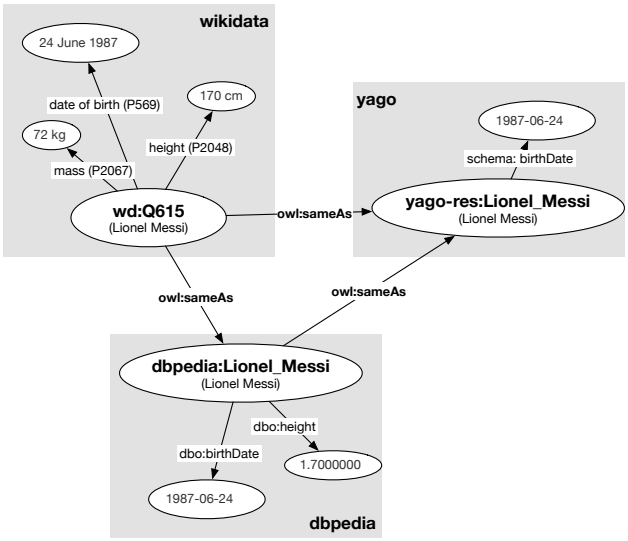


Figure 5.1: Multi-KG answer to query about Lionel Messi (snippet)

In addition, data stored in KGs are manually entered by users/experts or automatically extracted from Web documents and entered by computer programs at different points in time results in different values of the same objects, not mentioned human errors. As a result, data collected from multiple different KGs may contain conflicting facts. In the question answering task, a system is required to select and provide reliable values (objects) to users. In a nutshell, the requirement is to compare triples obtained from multiple KGs

and identify a single object – among all objects from the retrieved RDF triples – that is the most representative. In the process of comparing the triples, we consider three sources of inconsistency and uncertainty:

- properties used in different KGs have different names and not always they have the same meaning, some degree of equivalence needs to be determined;
- objects of triples, i.e., values that are subject of comparison are not always identical, some degree of similarity between should be calculated;
- sources of information, i.e., knowledge graphs could be linked with different levels of trustworthiness.

The proposed process of determining the most representable item (object) among a set of RDF triples resemble a process of identifying medoids (representative objects of a data set or a cluster within a data set) – we look for an item with the highest degree of similarity to other items from the set.

5.1.2 Method Overview

Let us have a set of triples collected from multiple KGs. For example, we query KGs [95] for a birthday of Lionel Messi (Figure 5.1), and obtain a set of triples with *Lionel Messi* as the subject. However, names of the property *birthDay* could be different (vocabulary dependent), as well as dates themselves could be of various formats and values. In general, the set is composed of triples with the same subject, but with different properties and objects

$$RdfT = \{\langle s, p_i, x_i\rangle \mid i = 1, \ldots, N\}. \tag{5.1}$$

Our goal is to select a single x_i, a date of birth in the above example, that is a reliable value in the set *RdfT*. For that purpose, we introduce a measure called *veracity*. It is a measure that indicates how similar a single item is to other items. The selected item is identified by finding a maximum over all elements of *RdfT*

$$x^* = argmax_{x_i}(veracity(x_i)). \tag{5.2}$$

The measure takes into account mentioned above three sources of inconsistency and uncertainty. It is calculated using the following formula

$$veracity(x_i) = \left[EQ(p, p_i) * \sum_{\substack{j=1 \\ j \neq i}}^{N} sim(x_i, x_j) \right] * T_i \tag{5.3}$$

where $EQ(p, p_i)$ is a degree of equivalence between p_i of the RDF triple with x_i as its object and the property p that is treated as the reference property determined during generation of the query [95]; $\sum sim(x_i, x_j)$ represents similarity between x_i and all x_j of the triples from *RdfT*, and finally T_i is a degree of trustworthiness of the KG from which x_i is obtained.

Based on calculated values of veracity measures for all elements of the set *RdfT* we can determine a level of confidence in the value selection

$$conf(x^*) = \frac{\sum_{i=1}^{n} veracity(x_i | x_i = x^*)}{\sum_{j=1}^{n} veracity(x_j)}. \tag{5.4}$$

It reflects a level of consistency in the obtained answers. If all x_i's are the same regardless of their veracity, the level of confidence is 1.0 – all queried KGs agree on the answer. On the other hand, if KGs do not agree on the answer – different values of x_i as well as of $EQ(p, p_i)$ and of T_i – the confidence value reflects that and drops below 1.0.

5.2 Property Equivalence

In this section, we provide a short description of the approach we use to determine equivalence between properties, i.e., $EQ(p, p_i)$ required for calculating veracity.

Let us have $KG_1, KG_2, ..., KG_n$ as data sources (Knowledge Graphs) whose corresponding trustworthiness are $T_1, T_2, ..., T_n$. One of them – say KG_1 without loss of generality – is selected as a reference KG – KG_r. It is a primary KG used in our query process [95]. A subject (in a sense of RDF triple) of a query, Lionel Messi in our example from Section 5.1.1, is mapped to items on other KGs, for the purpose of running queries there, via a property *owl:sameAs*. This property is defined by Web Ontology Language (OWL), to indicate that two

items refer to the same thing. In such case, we denote p and p^i are properties of KG_r and KG_i, respectively.

Previously, we have introduced a simple approach to determine degrees of equivalences between properties defined by different vocabularies [96]. In this paper, we propose its modified version

$$\begin{aligned} EQ(p, p') = \alpha * labelSim(p, p') + \\ + (1 - \alpha) * tripleSim(p, p') \end{aligned} \tag{5.5}$$

where $'$ means $i = 2, \ldots, n$.

The proposed equivalence degree is a linear combination of label-based similarity and triple-based similarity between the two properties. The former is calculated as follows

$$labelSim(p, p') = cosine(labelV(p), labelV(p')) \tag{5.6}$$

where $labelV(p)$ is a function that returns a vector embedding of p's label.

The later is computed as follows

$$tripleSim(p, p') = \frac{\sum_{j=1}^{M} objectSim(O_j, O'_j)}{M} \tag{5.7}$$

where O_j and O'_j are non-empty sets of objects (in a sense of RDF); M is the number of triple pairs $\big(\langle s_j, p, O_j\rangle \in KG_r;\ \langle s'_j, p_i, O'_j\rangle \in KG_i\big)$ such that $\langle s_j, owl{:}sameAs, s'_j\rangle$; and $objectSim(O_j, O'_j)$ is

$$objectSim(O_j, O'_j) = max(sim(x, y)) \tag{5.8}$$

where $sim(x, y)$ is a similarity degree between $x \in O_j$ and $y \in O'_j$ and is computed w.r.t their datatype (see Section 5.3).

5.3 Data Similarity

Objects of RDF triples could be of different datatype. The most common ones are DATE, NUMBER, STRING, and URI. The similarity between two values for each of these datatypes is calculated differently. Besides, if any value is missing we denote it as 'unknown'. The similarity of an 'unknown' with another value regardless of its datatype is always qual to 0.

To accommodate different datatypes, we proposed the following similarity measures for each of them. For DATE, we have

$$dSim(x_i, x_j) = \frac{len(commonStr(str(x_i), str(x_j))}{8} \tag{5.9}$$

where *commonStr* is a function that returns a leftmost longest common string, while *str* is a function that returns a string representation of a DATE in the format 'DDMMYYYY'.

The similarity between two NUMBERs is given by following formula with $\alpha \geq 0$.

$$nSim(x_i, x_j) = \begin{cases} 1.0 & \text{if } x_i = x_j = 0 \\ 1 - \frac{|x_i - x_j|}{|x_i| + |x_j|} & \text{if } x_i * x_j > 0 \\ 1 - \frac{|x_i - x_j|}{|x_i| + |x_j| + max(x_i, x_j)} & \text{if } x_i * x_j < 0 \\ \frac{\alpha}{|x_i| + |x_j|} & \text{if } x_i * x_j = 0. \end{cases} \tag{5.10}$$

If x_i and x_j are two STRINGs whose vector representations are available, their similarity is computed using a vector similarity metric such as cosine

$$sSim(x_i, x_j) = cosine(V(x_i), V(x_j)) \tag{5.11}$$

where V is an embedding vector of a string. If x_i and x_j are two STRINGs whose vector representations are not available, their similarity is computed using a string similarity metric to x_i and x_j such as Levenshtein distance, Jaro distance, or Smith-Waterman distance.

If x_i and x_j are two URIs, their similarity degree is calculated as follows

$$uSim(x_i, x_j) = \begin{cases} 1.0, & \text{if } \langle x_i, owl : sameAs, x_j \rangle \\ 0, & \text{otherwise} \end{cases} \tag{5.12}$$

5.4 Trustworthiness

The comparison of answers obtained using different KGs should include a measure indicating a degree to which we trust a given KG. It means that a process of identifying trustworthiness of KGs is quite important. In this section, we look at processes of initialization of levels of trustworthiness, as well as their update.

5.4.1 Initialization – Saaty-based Method

Given a collection of data sources (Knowledge Graphs) $KG_1, KG_2, ..., KG_n$ we want to estimate their trustworthiness $T_1, T_2, ..., T_n$. The method is introduced by Saaty in 1980s [88], and the process of estimation is as follows.

For each pair of KGs – KG_i and KG_j – an expert, a designer, or a user is asked to provide her pair-wise trust in them using a scale with values from 1 and 7 (a scale of 5, or 9 levels can be used as well). If KG_i is trusted more than KG_j then she is willing to assign a higher value, say 6 or 7 to the entry (i, j) in a so-called reciprocal matrix R. A low value, say 3 or 2 is assigned otherwise. The value of 1 indicates that KG_i and KG_j are equally trusted. The inverse of the number in the entry (i, j) is automatically assign to the entry (j, i).

$$R = \begin{pmatrix} 1 & r_{1,2} & \cdots & r_{1,n} \\ \frac{1}{r_{1,2}} & 1 & \cdots & r_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{1}{r_{1,n}} & \frac{1}{r_{2,n}} & \cdots & 1 \end{pmatrix}$$

Next, the maximal eigenvalue and its corresponding eigenvector are computed. The eigenvector associated with the largest eigenvalue is then the estimated trustworthiness.

The constructed reciprocal matrix R has an important property of transitivity. It means that for all indexes i, j, and k $r_{ij} * r_{jk} = r_{ik}$. This property grantees the consistency in evaluation. Also, due to characteristic of R, its largest eigenvalue λ_{max} is never less than n. The following ration is suggested in [84]

$$\vartheta = \frac{\lambda_{max} - n}{1 - n}$$

as an index of data inconsistency. If ϑ is less than 0.1, the estimation process is sought to be consistent. Whereas a higher value of ϑ calls for a rerun of the process.

The aforementioned one-expert priority method of trustworthiness estimation is not free of bias. To alleviate this, multiple experts can be asked to compare the trustworthiness of KG_i and KG_j using the same scale.

5.4.2 Initialization – Property-Equivalence Method

Another approach that can be used to initialize values of trustworthiness is based on determined values of equivalence of properties of two different KGs.

Assume that our reference Knowledge Graph KG_r has N properties $p_1, p_2, ..., p_N$. The trustworthiness of another Knowledge Graph is initialized with respect to KG_r following the formula

$$T_i = \frac{\sum_{j=1}^{N} EQ(p_j, p_i)}{N} \tag{5.13}$$

Please, note that p_i is a KG_i property that is the most equivalent to p_j according to $EQ(p_j, p_i^k)$ for $k = 1$ to M (M is the number of properties of KG_i). Again, KG_r is the reference KG.

5.4.3 Trustworthiness Update

The trustworthiness T_i of KG_i is updated every time a reliable data (x^*) is determined . The update process follows Algorithm 2.

Algorithm 2 Updating trustworthiness

```
1: procedure UPDATING(x* : trueValue, r : reward, n : numberOfSources )
2:     count ← 0
3:     simTotal ← 0
4:     for i = 1 to n do
5:         simTotal ← simTotal + sim(x_i, x*)
6:         if x_i = x* then
7:             count ← count + 1
8:         end if
9:     end for
10:    if count ≠ n then
11:        for i = 1 to n do
12:            if x_i = x* then
13:                T_i ← T_i + r/count
14:            else
15:                T_i ← T_i − r ∗ (1 − sim(x_i, x*))/(n − simTotal)
16:            end if
17:        end for
18:    end if
19: end procedure
```

The main idea of the algorithm is that whenever a reliable value is selected we update trustworthiness of KGs if they provide conflicting values. We in-

crease the trustworthiness of the KG_i if it provides x_i that is identical to x^* but decrease its trustworthiness an amount that is proportional to degree of dissimilarity between x_i and x^*. Overtime, the data source KG_i will be promoted more credits if it provides more values that is equal to true value.

5.5 Case Study

In order to evaluate a QA system over multiple KGs, we need a dataset that is to 'tied' to any KG, i.e., KG-independent. The existing benchmarks are KG specific: Free917 [12] and WebQuestions [7] are *Freebase*-based; versions of QALD[1] benchmark contain questions to be answered over either *DBpedia* or *Wikidata*; while the LC-QuAD [3] contains a set of complex questions that can be answered over *DBpedia*.

Therefore, to illustrate the usefulness of the proposed data fusion method for our QA system, we constructed a KG-independent dataset. It is based on the 2018 FIFA World Cup Russia List of Players downloaded from www.fifa.com. The dataset contains information about 736 players from 32 national teams. This dataset is created as an KG-independent, and is treated as a 'golden standard' against which we compare the results of the data fusion experiments.

The collected data for the data fusion experiments come from three of the largest cross-domain KGs in Linked Open Data (LOD) cloud. They are *DBpedia*, *Wikidata*, and *YAGO*. *DBpedia* contains close to 2 billion pieces of information (RDF triples) out of which 400 million were extracted from the English edition of Wikipedia. It is connected with *Wikidata*, *YAGO*, *GeoNames*, etc. via around 50 million RDF links. *DBpedia* is considered 'the hub' of the LOD cloud.

A very small snippet of *DBpedia* is shown in Figure 5.2. It includes RDF triples describing a few players of the football club *FC Barcelona*. As it can be seen, there are nodes that are connected to multiple other nodes. We can imagine that a single KG is highly interconnected.

[1] http://qald.aksw.org/

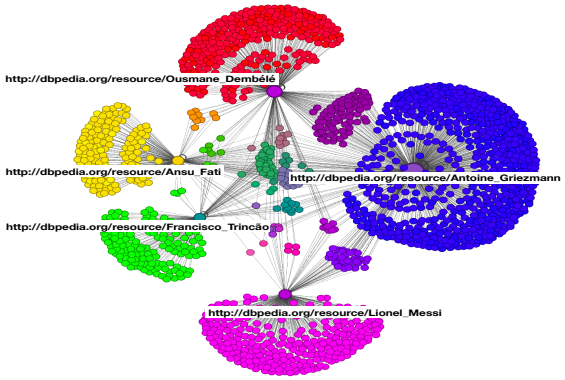


Figure 5.2: RDF triples representing players of *FC Barcelona* (*DBpedia*)

Table 5.1: Number of datapoints collected from KGs

Knowledge Graph	Birth Date	Height	Weight	Club
Wikidata	736	680	612	736
DBpedia	736	735	–	729
YAGO	717	–	–	735

5.5.1 Data Collection

The three best-known and largest cross-domain KGs are *Wikidata*, *DBpedia*, and *YAGO*. Each of them contains billions of RDF triples.

The QA system queries each KG for player's *Birth Date*, *Height*, *Weight*, and *Club* based on the player's team and name as included in the List of Players. The details of retrieved data are in Table 5.1.

5.5.2 Analysis of Retrieved Data

We evaluate the correctness of the retrieved data in terms of how accurately it matches the data provided on the List of Players. The data about a player is correct if there is an exact match with the information provided by the List. For a case of multiple football clubs, we consider the information as correct if the club on the FIFA list is included in the retrieved data. If a club name is available as a string its correctness is assessed using Levenshtein distance (the distance has to be less than three characters). If a club is identified via URI then it has to be connected via *owl:sameAs* with the URI of club on the List. Table 5.2 shows the results.

Table 5.2: Correct answers provided by each KG

Knowledge Graph	Birth Date	Height	Weight	Club (string)	Club (URI)
Wikidata	723	339	157	179	525
DBpedia	729	548	–	253	580
YAGO	708	–	–	171	508

Table 5.3: Conflicting information between KGs

Birth Date	Height	Club (URI)
13	292	24

Table 5.4: Correct answer for fused data; KGs' trustworthinesses not considered

Method	Birth Date	Height	Club (string)	Club (URI)
Exp_0	727	538	195	530

If we compare the content of Table 5.2 with one of Table 5.1, we see that retrieved information from KGs matches quite well data regarding *Birth Date* and *Clubs*, yet it shows many incorrect entries for *Height*.

To be accurate, we examine inconsistencies between pieces of information collected from the KGs. Table 5.3 includes numbers of conflicts regarding information about players' *Birth Dates*, *Heights* and *Clubs* for which they played when World Cup 2018 took place.

5.5.3 Fusion of Retrieved Data

We start experiments with fusing the data without considering trustworthiness of KGs – let us call the experiment as **Exp_0**. Table 5.4 shows the obtained results.

In the next experiments, we integrate the retrieved data – *Birth Date* and *Club* from all three KGs; and *Height* from *Wikidata* and *DBpedia* – with three different methods of initialization of trustworthiness. For the first and simplest method (**Exp_1**) we assign a value of 1.0 as the degree of trustworthiness for each KG. In the second case, (**Exp_2**), the values of trustworthiness are initialized using the Satty's priority method. In this approach, we use the scale

Table 5.5: Correct answers for fused; different methods of initialization of trustworthiness

Method	Birth Date	Height	Club (string)	Club (URI)
Exp_1	724	339	179	525
Exp_2	729	549	253	629
Exp_3	729	549	253	629

of 7 to compare KGs pair-wise. A reciprocal matrix R used in our experiment is

$$R = \begin{pmatrix} 1 & \frac{1}{3} & 5 \\ 3 & 1 & 7 \\ \frac{1}{5} & \frac{1}{7} & 1 \end{pmatrix}.$$

The maximal eigenvalue equals $\lambda_{max} = 3.06$, which is slightly higher than the reciprocal's dimension. The corresponding eigenvector is equal to [0.39, 0.91, 0.10] and it represents the trustworthiness values of *Wikidata*, *DBpedia*, and *YAGO*, respectively.

The third method of initialization of KG's trustworthiness (**Exp_3**) uses degrees of equivalence between properties. We selected *DBpedia* as the primary KG that it is *de facto* a hub for Linked Open Data [64]. The average values for the property-equivalence between the hub and *Wikidata*, and the hub and *YAGO* are 0.62 and 0.43. Thus, the values of trustworthiness for *Wikidata*, *DBpedia*, and *YAGO* are 0.62, 1.0, 0.43.

With such an initialization of trustworthiness, we perform three experiments with the proposed method for data fusion. The evaluated correctness of the fused data is shown in Table 5.5.

5.5.4 Discussion

Let us analyze the obtained results. The QA system performs slightly better when we fuse data from multiple different KGs than it does based on the data collected from any single KG, Tables 5.2 and 5.5. The tables also show that the system performs much better for the case of answering questions related to clubs of players when clubs are represented by URIs than by their names (strings).

An interesting observation is that 'static' data, such as *Birth Date* are easier to be correct while 'dynamic' data such as *Height* and *Weight* are the most inconsistent, Tables 5.1 and 5.5.

It seems that the nonuniform assignment of trustworthiness to KGs produce much more accurate answers than the uniform one, Tables 5.5 and 5.4. When we compare the results of **Exp_0** with the results of **Exp_1** we can say that the numbers of correct answers when fused without taking into consideration of KGs' trustworthiness are approximately equal to ones when KGs' trustworthiness are used but equally initialized. Another interesting observation can be done when the results of **Exp_2** are **Exp_3** are analyzed: the results much better when compared with the ones obtained for **Exp_0** and **Exp_1**, and they are the same. This indicates that both initialization methods are equally effective and either of them can be used – it would depend if any experts are available or not. This also suggests that trustworthiness play an imporant role in determining a true value from ones that are provided by multiple sources of data.

5.6 Related work

5.6.1 Data Fusion Methods

Data fusion is the process of finding a true value from conflicting values provided by different sources [25], or the process of fusing multiple records representing the same real-world object into a single, consistent, and clean representation [9].

Data fusion is considered firstly mentioned by Dayal in his paper published in 1983 [9]. A simple approach to data fusion is majority voting with an assumption of equally reliable sources. According to this approach, choose the value made by the most sources. More advanced techniques often combine the popularity or voting of candidate values with the quality (reliability/trustworthiness/authority) of their sources.

Many approaches have been introduced for estimating the quality. One of the most cited approach is of iterative updating trustworthiness. Two influ-

ential methods that exploit the link structure of Web page are PageRank [80] and Authority-Hub analysis [59]. PageRank takes advantage of the link structure of the Web to produce relative importance of web pages to help search engines and users quickly make sense of the vast heterogeneity of the World Wide Web. It starts with an rank initialization, then iterates a eigenvector calculation based on backlinks a pages until convergence criteria is met. With some shared similarities with PageRank, Authority-Hub estimates the importance (authority) of information sources (such as web pages) by utilizing both out-degree and in-degree hyperlinks instead of solely in-degree ones. Another difference is that instead of analyzing a full graph (containing all pages), author focuses on a subgraph: a collection of highest ranked hyperlinked pages from a text-based search engine related to a specific search topic to reduce computational cost. A more complicated framework proposed by Pasternack et al. [82] incorporate many more factors such as the uncertainty in the information extraction of claims from documents, attributes of the sources, the degree of similarity among claims, and the degree of certainty expressed by the sources into the fact-finding process was.

Yin et al. introduced an iterative computational method for finding true facts from conflicting multi-website information by which at each iteration the probabilities of facts being true and the trustworthiness of websites are inferred from each other. The algorithm is called TRUTHFINDER that was claimed that can select better trustworthy websites than authority-based search engines such as Google on their experiments [116].

5.6.2 Data Fusion of RDF Data

Mendes et al. introduced Sieve that is a part of a Linked Data Integration Framework. Sieve consists of a data quality assessment module and a data confusion one. However, both are user-defined tasks. In particular, users select quality indicators such as "Recency", "Reputation" and scoring functions such as "TimeCloseness", "Preference" for quality assessment; users also chosen actions such as ignoring conflicts("PassItOn"), filtering out information ("Filter") and fusion functions such as "KeepSingleValueByQualityScore",

"PickMostFrequent" for data fusion [74]. Dong and Srivastava implemented techniques generating snapshot explanations for the data-fusion decision [26]. Liu et al. proposed triple-embedding similarity-based algorithms for resolving conflicts when fusing RDF data, but not much detail was provided [67]. S. Thoma et al. proposed an entity-centric RDF data fusion through hierarchical clustering, the representative selection, finally ranking simply by the number of sources that support them. However, contradicting information has not yet addressed, as authors admitted [94]. In an attempt to integrate different versions of RDF graph to generate entity summaries, Tasnim et al. used some fusion policies such Union, Subproperty, Authority Graph, Policy Summary Policies to resolve conflicts [93].

5.6.3 Data Fusion and KB-based QA Systems

Among QA systems that have been developing so far, only few, for example [21], [61], [62], [81], are able to collect and merge answers from multiple different sources. However, authors of the systems, for instance Park et al. [81] and Diefenbach [21], provided very little information about how information was fused. Ko et al. [61], [62] applied a probabilistic graphical model that estimates the joint probability of correctness and correlation of all answer candidates to produce accurate and comprehensive answers collected from a large collection of documents. Lyu et al. [72] proposed a user-expertise specific learning framework that encodes latent user vectors into the answer representation by hierarchical attention mechanisms within Long-Short Term Memory (LSTM) network for answer selection in a community question answering(CQA). Lui et al. [68] presented an online system, called SOLARIS, that starts with returning the answers from the first probed source, refreshes the answers as it probes more sources and terminates probing when it gains enough confidence (expected, maximum, and minimum probability calculated based on retrieved data, accuracy of the sources, and copying between the sources) that data from the unprocessed sources are unlikely to change the returned answers.

5.7 Conclusion

The exceptions of better utilization of information stored on the Web in a semantically rich format of RDF create needs to build QA systems able to retrieved information and data from multiple different Knowledge Graphs.

In this chapter, we proposed a data fusion method that can determine the most reliable (a true value) data that stands for a collection of data points retrieved from multiple KGs. We also proposed and investigated methods for the initialization of trustworthiness in KGs.

Our experiments show that the trustworthiness of data sources/KGs should be included in data fusion processes. It seems that the trustworthiness should be updated over time based on how accurate information each KG provides.

We also investigated various approaches for trustworthiness initialization. The expert-based one is simple, universal, and computationally efficient, yet subjective. On the other hand, the data-based method is time-consuming but objective. However, when collecting data from multiple KGs with various vocabularies the process of determining equivalent properties is inevitable, thus the degrees of equivalence are already computed and can be utilized.

Chapter 6

Data summarization

The increased popularity of Linked Open Data (LOD) and advances in Natural Language Processing techniques have led to the development of Question Answering Systems (QASs) that utilize Knowledge Graphs as data sources. QASs perform well on simple questions providing precise and concise answers. Yet, most of them cannot process answers that contain a large volume of numerical values and are not able to provide users with answers in a human-friendly format.

In this chapter, we propose a user-defined method for constructing linguistic summarization of multi-feature data. It selects suitable summarizers and quantifiers and works with linguistic constraints imposed on the data. The method utilizes definitions of linguistic terms provided by users with an easy and simple graphical interface. Additionally, we introduce a Context-based User-defined Weighted Averaging (CUWA) operator that allows users to determine an average over data that satisfy multiple constraints that constitute defined by user's context. We include several illustrative examples.

6.1 Human-friendly Output Interface

In order to make the output of our question answering system human friendly, we have developed *Human-friendly Output Interface* that contains a method for context-based averaging of data, and a procedure for human-tuned processing of data in order to generate its overview.

At first (Section 6.2), we present a simple CUWA operator. It provides

the user with the ability to define a set of multi-feature (multi-dimensional) conditions that are imposed on data being processed. This multi-dimensional context is created on data attributes and uses linguistic terms provided by the user. CUWA allows to determine an average value or find min/max values of a single attribute on data points satisfying the terms of context.

Following that, we introduce and describe the User-defined Linguistic Summarizer (Section 6.3). The process is based on 'fitting' the results of a query into a protoform [117]:

$$Q\ F_C(\mathcal{O})\ are/have\ S \tag{6.1}$$

where $\mathcal{O}$ represents data objects obtained as the result of a query, S is a summarizer, Q is a quantifier, and F_C is a linguistic constraint representing a context. The users define linguistic terms that reflect their perceptions of terms in the domains of selected attributes. These terms constitute a set of summarizers. The procedure is developed to select the most suitable term/summarizer S representing the query answer data $\mathcal{O}$. Similarly, a suitable quantifier Q is automatically selected from a set of quantifiers provided by the users.

We would like to highlight, that linguistic terms such as TALL, YOUNG, HIGH, and quantifiers like FEW, MOST are imprecise concepts that are widely used in everyday conversations. However, meanings of such linguistic terms are invariably user-dependent. There are no fixed or predefined fuzzy sets that will fully fit perceptions of all users. To address this issue in the light of the proposed human-friendly output interface, we utilize the *LingTeQA* system's GUI interface dedicated to defining membership functions representing linguistic terms. The interface enables users to draw shapes of membership functions [113].

6.2 Context-based User-defined Weighted Averaging Operator

During data analysis processes, quite often there are situations when a user would like to learn more about data in a specific context. For example, given

an IQ test dataset one could ask *'what is the average IQ score of YOUNG people?'*. There are a few possible answers to such a question: if an average IQ score is calculated over all people, we ignore the context 'YOUNG'; if a crisp condition for selecting YOUNG people is used, for example, $AGE < 25$, many people whose age is 'just after' YOUNG are left out. It seems the calculation of IQ score should reflect – via some kind of weights – a level of satisfying the concept YOUNG by each individual in the group.

6.2.1 CUWA Definition

Let us assume a scenario where the *LingTeQA* system answers the user's question and as a result we obtain a multi-dimensional (multi-attribute) data. It means, the collected data is a set of n objects $\mathcal{O} = \{o_1, o_2, ..., o_n\}$, with each object described by m attributes: $A^1, A^2, ..., A^k, ..., A^m$, where

$$A^k = \{a_1^k, a_2^k, ..., a_n^k\}. \tag{6.2}$$

For our exemplary question, $\mathcal{O}$ is a set of people, A^1 represents their IQ scores, while A^2 is the attribute AGE. The collected data could be used to answer such questions as: *'what is the average IQ score of YOUNG people?'* or *'what is the average AGE of people of HIGH IQ score?'*. Both of these questions identify a context: 'YOUNG people' in the case of the first question, and 'HIGH IQ score' in the case of the second one. Each of them is related to a single attribute and can be represented as a linguistic term. Once individuals satisfy the context to a required degree the values of the other attribute, IQ score or AGE in our example, can be averaged.

In general, a context can be defined based on multiple attributes. In such a case, a logical combination of linguistic terms is employed. Because, data objects can satisfy a given context to a different degree, the user also provides *Context Satisfaction Level (CLS)* that represents the user's minimum level of satisfying the context terms by data.

In summary, to calculate a context-based average, the user identifies an attribute that should be averaged, as well as fuzzy sets corresponding to linguistic terms of the context and operations performed on them. Based on the

latter, a fuzzy set F_C representing a context is constructed and applied to the considered data set.

Let $f_C = \left[f_{C,1}, ..., f_{C,n}\right]$ is a sequence of degrees of satisfaction of the context F_C by data objects. We create a set $\mathcal{P}$ of indexes of data objects that satisfy the CSL, i.e.,

$$\mathcal{P} = \{k \mid f_{C,k} > CSL\} \tag{6.3}$$

Now, we calculate an average value of an attribute A^m in the context F_C using the following equation:

$$CUWA_{CSL}(a_1^m, ..., a_n^m, F_C) = \sum_{j \in \mathcal{P}} w_j * a_j^m \tag{6.4}$$

where w_j, for j belongs to $\mathcal{P}$, is computed as

$$w_j = \frac{f_{C,j}}{\sum_{i \in \mathcal{P}} f_{C,i}} \tag{6.5}$$

Eq. 6.5 guarantees that $w_j \in [0, 1]$ and $\sum_{j \in \mathcal{P}} w_j = 1$. Now, Eq. 6.4 can be rewritten as

$$CUWA_{CSL}(a_1^m, ..., a_n^m, F_C) = \frac{1}{\sum_{i \in \mathcal{P}} f_{C,i}} \sum_{j \in \mathcal{P}} f_{C,j} * a_j^m \tag{6.6}$$

6.2.2 CUWA Extension

The presented above CUWA calculates an average value over an attribute. It can be easily extended to handle situations that require determining a min or max value of the attribute. The following equations are modifications of CUWA that address such requirements:

$$CUWA_{CSL}^{min}(a_1^m, ..., a_n^m, F_C) = min\{a_j\} \tag{6.7}$$

$$CUWA_{CSL}^{max}(a_1^m, ..., a_n^m, F_C) = max\{a_j\} \tag{6.8}$$

where j belongs to $\mathcal{P}$.

Another extension, generalization, of *CUWA* that involves applying an arbitrary function on values of the attribute to be averaged. In such a case, a more generic version of Eq. 6.4 is

$$CUWA_{CSL}(a_1^m, a_2^m, ..., a_n^m, F_C) = \sum_{j \in \mathcal{P}} w_j * g(a_j^m) \tag{6.9}$$

where g is a strictly continuous monotonic function. For $g(b) = b$, CUWA becomes a Weighted Averaging operator, while for $g(b) = b^2$, CUWA becomes a Weighted Quadratic Averaging operator.

6.3 User-defined Linguistic Summarizer

Quite often, a large numerical data is obtained as the result of asked questions. In such a situation, the user does not want to analyze the data by herself – the focus is not on the details but on some kind of overview – summary – of the obtained data. To address this need, we propose a multi-dimensional summarization process. We provide a procedure that allows for summarizing numerical data over a number of different attributes (dimensions), and in the context specified by the user. The summarization process results in a linguistic summary of data that is in the form of a 'customized' protoform [117].

$$Q\ \mathcal{O}\ are/have\ S \tag{6.10}$$

or

$$Q\ F_C(\mathcal{O})\ are/have\ S \tag{6.11}$$

where $\mathcal{O}$ is a set of the obtained data objects, S is a summarizer: a single or compound/complex linguistic term; Q is a quantifier; F_C is a linguistic constraint. Specific instances of S and Q in the context of F_C are chosen from sets of user-provided terms. The user defines the meanings of these terms via drawing shapes of membership functions [113].

Let us start with a description of the basic process of single-dimensional (one feature) summarization of data. Further, we generalize the procedure via introduction of context and multi-dimensional summarization.

6.3.1 Single-Feature Summarization

We assume, the user provides a number m of summarizers $S_1, S_2, ..., S_m$ represented by a set of the user-defined fuzzy sets on the attribute A^k. We name

this set of the fuzzy sets $\Phi_{A^k} = \{F_{A^k,1}, F_{A^k,2}, ..., F_{A^k,m}\}$. Our goal is to select a summarizer S_j that best describes the data objects based on the attribute A^k.

Selecting Summarizer

The first step focuses on determining a value of α-cut that is used to identify 'core' data objects that are the basis for selecting the most suitable summarizer/fuzzy set describing the data. We execute the following equation:

$$\alpha = min(hgt(F_{A^k,1}), hgt(F_{A^k,2}), ..., hgt(F_{A^k,m})) \tag{6.12}$$

where $hgt(F_{A^k,j})$ is the height of a fuzzy set $F_{A^k,j}$ over all objects from $\mathcal{O}$, and $hgt(F_{A^k,j}) > 0$ for $j = 1, 2, ..., m$.

Once, α-cut is determined, we create core files $Core_j$ for each fuzzy set $F_{A^k,j}$. The core file contains data objects o with a degree satisfying a fuzzy set $F_{A^k,j}$:

$$Core_j = \{o \in \mathcal{O} | F_{A^k,j}(o) \geq \alpha \; and \; F_{A^k,j} \in \Phi_{A^k}\} \tag{6.13}$$

Now, we are ready to select a summarizer. This process is done using a few steps.

Step 1: we identify indexes of dominant summarizers based on the size of their core files

$$I_S^1 = \{j_0 \mid j_0 = arg, max_j card(Core_j)\}. \tag{6.14}$$

If I_S^1 is a singleton, then the summarizer S_j whose index $j \in I_S^1$ is chosen as a summarizer. Otherwise, we perform Step 2.

Step 2: we use indexes of summarizers obtained in Step 1, and determine fuzzy sets with the largest supporting set

$$I_S^2 = \{p_0 \mid p_0 = arg, max_{p \in I_S^1} card(Supp(F_{A^k,p}))\}. \tag{6.15}$$

Again, if the I_S^2 is a singleton, then the selection of a summarizer is done, $S_p \mid p \in I_S^2$. Otherwise, we proceed to Step 3.

Step 3: we determine indexes based on the sum of degrees of satisfaction of data objects from the supporting set. Now, we have

$$I_S^3 = \{q_0 \mid q_0 = arg, max_{q \in I_S^2} \sum_{j=1}^{n} F_{A^k,q}(o_j)\}. \tag{6.16}$$

If the I_S^3 is a singleton, then the summarizer $S_q \mid q \in I_S^3$ is chosen. If it is not a case, we randomly select a summarizer whose index belongs to I_S^3. In summary, at the end of the selection process, S_j is the selected summarizer.

Selecting Quantifier

Once the summarizer S_j is selected, we go back to its corresponding fuzzy set $F_{A^k,j}$.

The set of $F_{A^k,j}$ has

$$c = card(Supp(F_{A^k,j})) \tag{6.17}$$

objects. Additionally, to summarize fuzzy sets, the user also provides a number of quantifiers $Q_1, Q_2, ..., Q_t$ and associated with them fuzzy sets $F_{Q,i}$ whose membership functions are defined on the interval $[1, n]$. Based on the provided fuzzy sets, we find an index of quantifier for which its membership level for c has the highest value. In particular, it identifies

$$I_Q = \{i \mid i = arg, max_t F_{Q,t}(c)\}. \tag{6.18}$$

As the final step, we instantiate the protoform

$$Q_i \ \mathcal{O} \ are/have \ S_j \tag{6.19}$$

with the identified summarizer on the attribute A^k and the selected quantifier. The truth (validity) value, T, of the summary is

$$T(Q_i \ \mathcal{O} \ are/have \ S_j) = F_{Q,i}(c) \tag{6.20}$$

where T is a number from the unit interval – the closer the value of T is to one, the more truthful the generated summary is.

6.3.2 Multi-Feature Summarization

The presented above procedure can be generalized to the case of multi-feature summarization. Here, we apply a compound summarizer that is a combination of two or more single summarizers using logistic operations of AND and/or OR. Without a loss of generality, we assume $S_{A^r,1}, S_{A^r,2}, ..., S_{A^r,p}$ as summarizers whose fuzzy sets are defined on the attribute A^r and are represented by $\Phi_{A^r} = \{F_{A^r,1}, F_{A^r,2}, ..., F_{A^r,p}\}$, and $S_{A^s,1}, S_{A^s,2}, ..., S_{A^s,q}$ are summarizers whose fuzzy sets defined on A^s are $\Phi_{A^s} = \{F_{A^s,1}, F_{A^s,2}, ..., F_{A^s,q}\}$.

Based on the user-provided terms and corresponding fuzzy sets, we construct their combination. For example, if AND is the connective between each pair of summarizers, we have

$$S_l = S_{A^r,i} \text{ AND } S_{A^s,j} \tag{6.21}$$

for $l = 1, 2, ..., p * q$ and $i = 1, 2, ..., p$; $j = 1, 2, ..., q$. This leads to the construction of the following compounded summarizers and their corresponding fuzzy sets

$$\Phi = \{F_{A^r,A^s,l} \mid F_{A^r,A^s,l} = \textit{T-norm}(F_{A^r,i}, F_{A^s,j})\} \tag{6.22}$$

Afterwards, the summarizer selecting process introduced in Section 6.3.1 is performed to identify a compound summarizer from the set of constructed fuzzy sets.

A compound summarizer can be also formed via ORing single ones. It would follow the same procedure with a replacement of a T-norm in Eq. 6.22 with a T-cornorm.

6.3.3 Summarization with Linguistic Constraint F_C

Another form of the protoform (Eq. 6.11), involves a linguistic constraint F_C. Let us have this constrain's fuzzy set to be defined on the attribute A^v as F_{A^v}.

At the beginning, we select data objects with nonzero-membership to the constraint F_C (manifested by the fuzzy set F_{A^v})

$$\mathcal{O}' = \{o_i \in \mathcal{O} \mid F_{A^v}(o_i) > CSL\} \tag{6.23}$$

for $i = 1, 2, ..., n$.

Then, we refine fuzzy sets representing summarizers by selecting only the ones of which data objects belong to the set $\mathcal{O}'$

Finally, the selection procedure (Section 6.3.1) is carried out to identify the most suitable summarizer.

6.4 Case Studies

The presented process of generating linguistic summarization has been applied to the answers obtained for a number of different user's questions. The provided examples illustrate the details of the process and the obtained summaries. We also include a simple case of applying *CUWA*.

6.4.1 Single-Feature Summarization

Let us have a set of ages of a group of individuals: {25, 13, 12, 19, 37, 25, 56, 45, 73} with a single attribute *Age*. We use the summarization process to generate a response to the question *"How old are people in the dataset?"*.

In this case, the user is asked to provide four linguistic terms (summarizers) along with their definitions that can be used to summarize the response: VERY YOUNG (S_1), YOUNG (S_2), MIDDLE-AGED (S_3), and OLD (S_4). The user uses *TiFS* [113] to draw shapes of associated membership functions $F_{Ag,1}$, $F_{Ag,2}$, $F_{Ag,3}$, and $F_{Ag,4}$. They represent the user's perception of these terms. They are shown, after cleaning and processing, in Fig 6.1.

Table 6.1 includes the membership levels of data objects from the *Age* file calculated based on the user-defined membership functions, as well as quantities required to determine a suitable summarizer: *hgts*, $|Core|$, $|Supp|$, and $\sum$.

Among all provided by the user summarizers S_2 has the highest cardinality of the —*Core*— equal to four. Therefore, the linguistic term YOUNG is selected. Next, the user is asked to provide a set of quantifiers. Also, the membership functions representing them – FEW, ABOUT A HALF, MOST – are defined using *TiFS* [113], Fig 6.2.

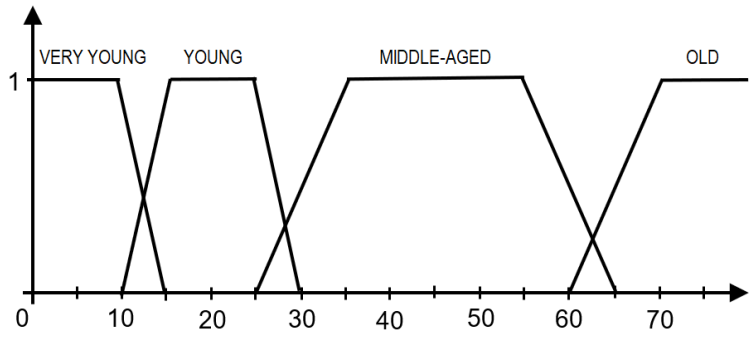


Figure 6.1: Membership functions describing linguistic terms – summarizers – defined on *Age*: VERY YOUNG ($F_{Ag,1}$), YOUNG ($F_{Ag,2}$), MIDDLE-AGED ($F_{Ag,3}$), and OLD ($F_{Ag,4}$)

Table 6.1: Membership levels of *Age* for linguistic terms, as well as calculated quantities used for selecting a summarizer

Age	Very young	Young	Middle-aged	Old
25	0	1	0.3	0
13	0.4	0.6	0	0
12	0.6	0.4	0	0
19	0	1	0	0
37	0	0	1	0
25	0	1	0.3	0
56	0	0	0.9	0.9
45	0	0	1	0
73	0	0	0	1
Hgt	0.6	1	1	1
$\|Core\|$	1	4	3	2
$\|Supp\|$	2	5	5	2
$\sum$	1.0	4.0	3.5	1.9

The value of $|Supp|$ for the summarizer YOUNG is equal to five. This translates into a degree of 0.8 for the quantifier ABOUT A HALF. Consequently, the answer to the user's question *"How old are people in the dataset?"* is *"ABOUT A HALF are YOUNG."*, with a truth value of 0.8 (Eq. 6.20).

6.4.2 Summarization of Data from Wikidata

Wikidata[1] is an open Knowledge Graph with nearly 60 million data items. It is a collection of items where each of them has a label, a description, and any

[1]https://www.wikidata.org/wiki/Wikidata:Main_Page

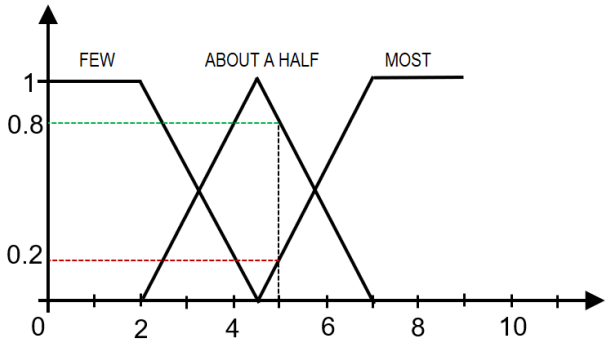


Figure 6.2: Membership functions describing quantifiers: FEW ($F_{Q,1}$), ABOUT A HALF($F_{Q,2}$), and MOST ($F_{Q,3}$)

number of aliases. Items are uniquely identified by a letter Q followed by a number, for example *Douglas Adams* has the identifier $Q42$. Items described with statements consist of a property and a value. In *Wikidata*, properties are marked with a letter P followed by a number – the property *educated_at* has the identifier *P69*.

Let the user asks our *LingTeQA* system a question *"How big are cities in Canada in terms of population?"*. Our system constructs a query, sends it to Wikidata Query Service, and retrieves a list of cities along with their population.

The system asks the user to provide four terms that would describe size of cities. The user provides the terms: SMALL, MEDIUM, LARGE, and VERY LARGE, together with their meanings by drawing four shapes using *TiFS* [113]. The corresponding membership functions as depicted in Fig. 6.3.

Based on the provided membership functions, *LingTeQA* computes their height, cardinalities of their core sets, and cardinalities of their supporting sets, Table 6.2.

Because its $|Core|$ is the largest with 286 cities, the summarizer SMALL (S_1) is chosen as the summarizer. The user also provides three quantifier: FEW, ABOUT A HALF, MOST. Their membership functions are given in Fig 6.4 .

The quantifier MOST is chosen with a membership level of 1.0 due to 286 cities that satisfy the term SMALL. As a result, *LingTeQA* provides the

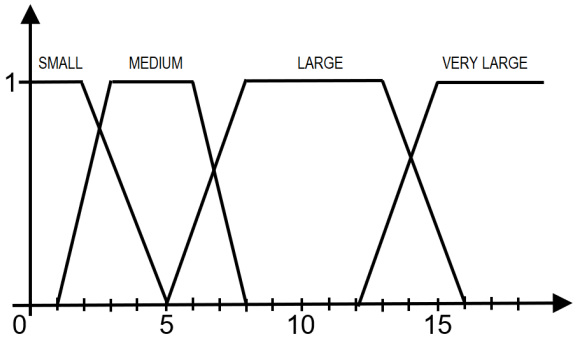


Figure 6.3: Membership functions describing linguistic terms – summarizers - defined on *Population*: SMALL ($F_{\text{Pop},1}$), MEDIUM ($F_{\text{Pop},2}$), LARGE ($F_{\text{Pop},3}$), VERY LARGE ($F_{\text{Pop},4}$) (in units of 100k)

Table 6.2: Required Quantities for Selecting Linguistic Term (Summarizer) Describing Population of Canadian Cities

	Small	Medium	Large	Very Large
Hgt	1.0	1.0	1.0	1.0
$\|Core\|$	274	9	2	2
$\|Supp\|$	286	39	8	3
$\sum$	282.3	19.2	4.0	2.9

answer: *"MOST cities in Canada are SMALL"* with the calculated truth value of 1.0 (Eq. 6.20)

Further, if the user asks a question *"What is the average number of people of SMALL cities in Canada?"* then the context is defined by the term SMALL with a *Context Satisfaction Level – (CLS)* of 0.5. Using the proposed *CUWA*, our system calculates an averaged value of 32,418 – so the system answers *"The average number of people in Canadian SMALL cities is 32,418."*

6.4.3 Multi-Feature Summarization

Now, we illustrate a scenario when summarizers are defined on two features. Let us define a set composed of a group of people who participated in an IQ test: *Test_Result* = { (5,158), (13,162), (12,104), (19,132), (37,127), (25,125), (56,142), (45,83), (73,137)}, where the first attribute is *Age* and the second is *IQ score*.

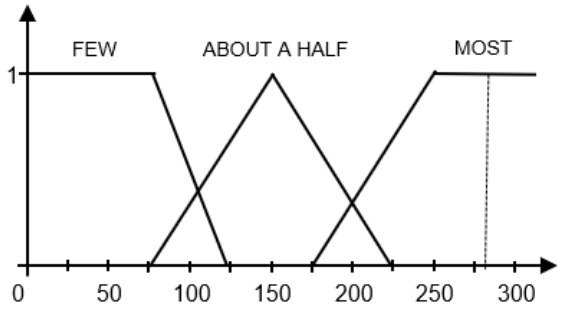


Figure 6.4: Membership functions describing quantifiers: FEW ($F_{Q,1}$), ABOUT A HALF($F_{Q,2}$), and MOST ($F_{Q,3}$)

The user asks a question *"How old and intelligent are people in the dataset?"*. This time, two sets of linguistic terms that describe people's age and intelligence are provided. The given set of linguistic terms and their membership functions for the attribute *Age* are shown in Fig 6.5, and for the attribute *IQ score* are in Fig 6.6.

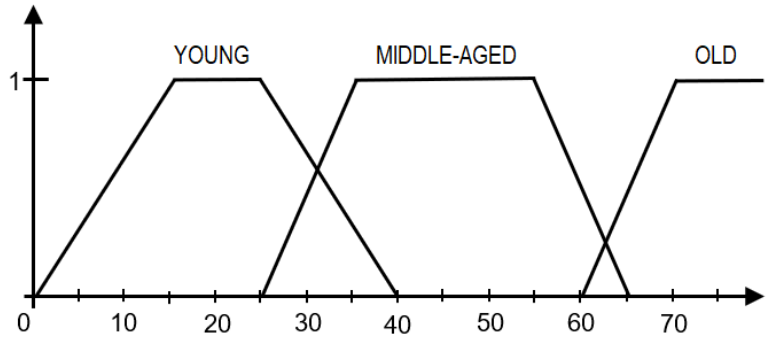


Figure 6.5: Membership functions describing linguistic terms (summarizers) of *Age*: YOUNG ($S_{Ag,1}$), MIDDLE-AGED ($S_{Ag,2}$), and OLD ($S_{Ag,3}$)

The values of membership levels calculated for the attribute *Age* are in Table 6.1, while for *IQ score* are in Table 6.3.

In total, there are nine combinations of linguistic terms for *Age* and *IQ score*. The membership level for each combination is obtained via aggregating individual levels using MIN as a *T-norm*. The obtained membership levels are included in Table 6.4. The combination $S_{\mathrm{Ag},1}\&S_{\mathrm{IQ},3}$ has the highest number of elements (5) for the set $|Core|$. Thus, the summarizer that contains YOUNG and HIGH_IQ is chosen.

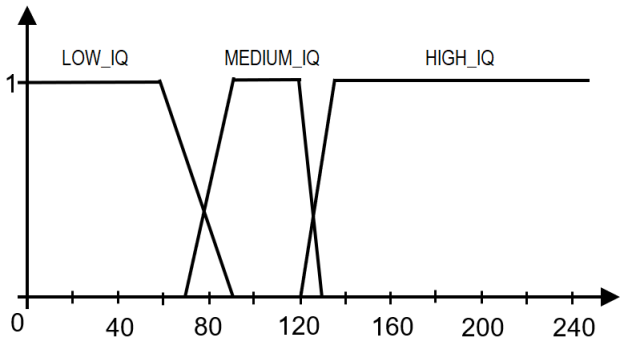


Figure 6.6: Membership functions describing linguistic terms (summarizers) for *IQ score*: LOW_IQ ($S_{\text{IQ},1}$), MEDIUM_IQ ($S_{\text{IQ},2}$), HIGH_IQ ($S_{\text{IQ},3}$)

Table 6.3: Membership grades of IQ score to linguistic terms

IQ score	LOW_IQ	MEDIUM_IQ	HIGH_IQ
158	0	0	1
162	0	0	1
104	0	1	0
132	0	0.4	0.8
127	0	0.7	0.5
125	0	0.8	0.3
142	0	0	1
83	0.2	0.4	0
137	0	0.2	1
$\lvert Core\rvert$	1	6	7
$\lvert Supp\rvert$	1	6	7

The quantifier (Q2) ABOUT A HALF with a membership level of 0.8 is selected based on the membership functions representing the three linguistic quantifiers as in Fig 6.2.

The answer to the user's question *"How old and intelligent are people in the dataset?"* is *"ABOUT A HALF are YOUNG and (have) HIGH_IQ."*

6.4.4 Single-Feature Summarization with Linguistic Constraint

In this example, the user wants to know *"How high IQ score have YOUNG people in the dataset?"* with the value of *Context Satisfaction Level* (*CSL*) equal to 0.5 for the linguistic constraint YOUNG. The dataset is the same as in the previous subsection, i.e., { (5,158), (13,162), (12,104), (19,132), (37,127),

Table 6.4: Membership levels of compound linguistic terms

No	$S_{\mathrm{Ag},1}$ $S_{\mathrm{IQ},1}$	$S_{\mathrm{Ag},1}$ $S_{\mathrm{IQ},2}$	$S_{\mathrm{Ag},1}$ $S_{\mathrm{IQ},3}$	$S_{\mathrm{Ag},2}$ $S_{\mathrm{IQ},1}$	$S_{\mathrm{Ag},2}$ $S_{\mathrm{IQ},2}$	$S_{\mathrm{Ag},2}$ $S_{\mathrm{IQ},3}$	$S_{\mathrm{Ag},3}$ $S_{\mathrm{IQ},1}$	$S_{\mathrm{Ag},3}$ $S_{\mathrm{IQ},2}$	$S_{\mathrm{Ag},3}$ $S_{\mathrm{IQ},3}$
1	0	0	1	0	0	0.3	0	0	0
2	0	0	0.9	0	0	0	0	0	0
3	0	0.8	0	0	0	0	0	0	0
4	0	0	0.8	0	0	0	0	0	0
5	0	0.2	0.2	0	0.3	0.5	0	0	0
6	0	0.5	0.3	0	0.3	0.3	0	0	0
7	0	0.	0	0	0	0.9	0	0	0.9
8	0	0.	0	0.2	0.7	0	0	0	0
9	0	0.	0	0	0	0	0	0	1
Hgt	0	0.8	1	0.2	0.7	0.9	0	0	1
$\|Core\|$	0	3	5	1	3	4	0	0	2
$\|Supp\|$	0	3	5	1	3	4	0	0	2
$\sum$	0	1.5	3.2	0.2	1.3	2.0	0	0	1.9

(25,125), (56,142), (45,83), (73,137)}. We assume, the sets of the user's linguistic terms for *Age* and *IQ score* and their membership functions are as shown in Fig 6.5 and Fig 6.6, respectively.

This time, the term YOUNG works here as a context constraint. It is applied to the attribute *Age* of objects from the set. The elements with a nonzero membership level are selected for further processing. Then, the values of the attribute *IQ scores* of these objects are used to determine levels of satisfaction of membership functions associated with terms LOW_IQ ($S_{\mathrm{IQ},1}$), MEDIUM_IQ ($S_{\mathrm{IQ},2}$), HIGH_IQ ($S_{\mathrm{IQ},3}$). These membership level are given in Table 6.5.

Based on the obtained values for quantities required for constructing a summary (Section 6.3) shown in Table 6.5, the summarizer HIGH_IQ is chosen because of its dominant cardinality of the set *Core*. Five object of the data set satisfy *CSL* of 0.5 for the constraint YOUNG people. The number five corresponds to the level 1.0 for the quantifier MOST (according to the definitions of linguistic quantifiers in Fig 6.7). This leads to selection of MOST as the quantifier for the summary.

The summary *"MOST (YOUNG people) have HIGH_IQ"* (with truth 1.0) is generated as the response to the question *"How high IQ score have YOUNG*

Table 6.5: Membership levels of YOUNG people for IQ score's linguistic terms

Age	IQ score	LOW_IQ	MEDIUM_IQ	HIGH_IQ
25	158	0	0	1
13	162	0	0	1
12	104	0	1	0
19	132	0	0	0.8
37	127	0	0.3	0.5
25	125	0	0.5	0.3
Hgt		0	1	1
$\|Core\|$		0	1	2
$\|Supp\|$		0	3	5
$\sum$		0	1.8	3.6

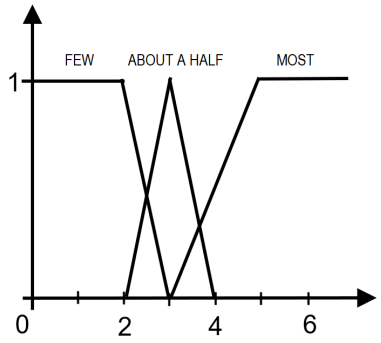


Figure 6.7: Membership functions describing quantifiers to filtered objects: FEW ($F_{\text{IQ},1}$), ABOUT A HALF ($F_{\text{IQ},2}$), MOST ($F_{\text{IQ},3}$)

Table 6.6: Membership levels for *Age* for linguistic terms

Age	Young	Middle-aged	Old
25	1	0.3	0
13	0.9	0	0
12	0.8	0	0
19	1	0	0
37	0.2	1	0
25	1	0.3	0
56	0	0.9	0.9
45	0	1	0
73	0	0	1

people in the dataset?".

6.4.5 Context-based Averaging using *CUWA* Operator

Let us take a question *"What is the average IQ score of YOUNG people?"* as an illustrative example of applying our *CUWA* operator. We use the membership level shown in Table 6.6 (column YOUNG) as a weighting vector for aggregating IQ score. The value for the *Context Satisfaction Level – CSL –* is set to 0.5. Thus, we obtain (Eq. 6.6):

$x = (1.0 * 158 + 0.9 * 162 + 0.8 * 104 + 1.0 * 132 + 1.0 * 125)/4.7 \approx 136$, where 4.7 is the sum of satisfaction levels for the contraint YOUNG.

The answer to the aforementioned question is *"The average IQ score of YOUNG people is 136."* This answer is very close to the one obtained for the question *"How high IQ score have YOUNG people in the dataset?"* form the previous subsection, i.e., *"MOST (YOUNG people) have HIGH_IQ."*

6.5 Related work

6.5.1 Generalized OWA Operators

The Generalized OWA operators, denoted as GOWA [111], is defined as

$$GOWA(a_1, a_2, ..., a_n) = \left(\sum_{j=1}^{n} w_j * b_j^{\lambda} \right)^{\frac{1}{\lambda}} \tag{6.24}$$

The GOWA operator is not only parameterized by weighting vector but also λ, which can take different values between -∞ and ∞, except 0, generating a different type of OWA aggregation. For example:

If $\lambda = 1$, the GOWA becomes the classic OWA operator.

If $\lambda = 2$, the GOWA becomes the ordered weighted quadratic average (OWQA)

If $\lambda = 3$, the GOWA becomes the ordered weighted cubic average (OWCA).

If $\lambda = -1$, the GOWA becomes the ordered weighted harmonic average.

If $\lambda = 0$, the GOWA becomes the ordered weighted geometric average (OWGA).

If $\lambda = -\infty$, the GOWA becomes the minimum aggregation.

If $\lambda = \infty$, the GOWA becomes the maximum aggregation.

Induced OWA operator (IOWA)

The IOWA [112] operator is an extension of the OWA operator that uses a more general reordering process of the information based on order inducing variables. It is formulated as follows:

$$IOWA(< u_1, a_1 >, < u_2, a_2 >, ... < u_n, a_n >) = \sum_{j=1}^{n} w_j * b_j \tag{6.25}$$

where b_j is the a_i value of the IOWA pair $< u_i, a_i >$ having the j^{th} largest u_i, u_i is the order-inducing variable, and a_i is the argument variable.

[112] proposed an alternative approach to ordering the arguments based upon using a function of the arguments rather than the arguments themselves in IOWA. In particular, authors proposed a 4-step iterative algorithm that is based upon the gradient descent in the back propagation method used for learning in neural networks.

IOWA can be extended as:

$$Quasi - IOWA(< u_1, a_1 >, ... < u_n, a_n >) = g\left(\sum_{j=1}^{n} w_j * g(b_j)\right) \tag{6.26}$$

where g is a strictly continuous monotonic function.

Probabilistic OWA operator (POWA)

The POWA operator [76] is an aggregation that uses probabilities and OWA operators in the same formulation.

$$POWA(a_1, a_2, ..., a_n) = \beta * \sum_{j=1}^{n} w_j * b_j + (1 - \beta) * \sum_{i=1}^{n} p_i * a_i \tag{6.27}$$

where b_j is the j^{th} largest of the a_i's, and $\beta \in [0, 1]$. $P = \{p_1, p_2, ..., p_n\}$ is a probabilistic vector such that $0 \leq p_j \leq 1$ and $\sum_{j=1}^{n} p_j = 1$.

Fuzzy Generalized Probabilistic Ordered Weighted Averaging Weighted Average (FGPOWAWA)

The proposed operator unifies the probability, the weighted average and the OWA operator with consideration of the degree of importance that each concept has in the aggregation [75].

The author further generalizes this approach by using quasiarithmetic means obtaining the quasi-arithmetic fuzzy POWAWA (Quasi-FPOWAWA) operator.

Fuzzy OWA (FOWA) operator is an extension of the OWA operator for uncertain situations where the available information can be assessed with fuzzy numbers Ψ [75].

$FOWA : \Psi^n \longrightarrow \Psi$ such that

$$OWA(\tilde{a_1}, \tilde{a_2}, ..., \tilde{a_n}) = \sum_{j=1}^{n} w_j * b_j \tag{6.28}$$

where b_j is the j^{th} largest of the $\tilde{a}_i$'s. $W = \{w_1, w_2, ..., w_n\}$ is a weighting vector such that $0 \leq w_j \leq 1$ and $\sum_{j=1}^{n} w_j = 1$.

The FOWA operator provides a parameterized family of aggregation operators that include the fuzzy maximum, the fuzzy minimum and the fuzzy average criteria, among others.

A FGPOWAWA operator of dimension n is a mapping FGPOWAWA:$\Psi^n \longrightarrow \Psi$ such that

$$FGPOWAWA(\tilde{a_1}, \tilde{a_2}, ..., \tilde{a_n}) = \left(\sum_{j=1}^{n} \hat{v}_j * b_j \right)^{\frac{1}{\lambda}} \tag{6.29}$$

where b_j is the j^{th} largest of the $\tilde{a}_i$'s, each argument $\tilde{a}_i$ is represented in the form of a fuzzy number and has an associated weight v_i such that $v_i \in [0, 1]$ and $\sum_{i=1}^{n} v_i = 1$, a probability p_i with $p_i \in [0, 1]$ and $\sum_{i=1}^{n} p_i = 1$, and .

The proposed operator (UWA) was the simplest form of a Fuzzy Generalized Unified Aggregation Operator (FGUAO) that is a further extended probabilistic weighted OWA (PWOWA). It is given as

$$UWA(a_1, a_2, ..., a_n) = \sum_{h=1}^{m} \sum_{i=1}^{n} C_h * w_i^h * a_i \tag{6.30}$$

where C_h is the degree of importance that each concept has in the aggregation with $C_h \in [0, 1]$ and $\sum_{h=1}^{m} C_h = 1$; w_i^h is the i^{th} weight of the h^{th} weighting vector W with $w_i^h \in [0, 1]$ and $\sum_{i=1}^{n} w_i^h = 1$

Many more variations of OWA have been proposed such as an Ordered Fuzzy Weighted Averages (OFWAs) and Ordered Linguistic Weighted Averages (OLWAs) are introduced by [104]. An Unbalanced Liguistic Ordered Weighted Average (ULOWA) is proposed by [48]. A Weighted OWA (WOWA) aggregation was presented by [95] where the weight w_j is generated using a function of sum of the importance weights associated with the j smallest data points.

6.5.2 Linguistic Summarization of Data

The ability to summarize data is an important means to grasp the meaning of the content of a large collection of data [110]. Linguistic summarization methods have been introduced by Yager, Zadeh and further pursued by Kacprzyk and Zadrożny, and others.

The classic linguistic summary, introduced by Yager [110], is a one-attribute linguistic summarizer that can be extended to cover more sophisticated summaries involving some confluence of attribute values.

Kacprzyk and Zadrożny have extended the work of Yager and Zadeh by developing new tools to handle more modalities in fuzzy querying, and linguistic data summaries [55]. In particular, they constructed a user interface add-on for of fuzzy querying – FQUERY for Microsoft Access – that was generating a linguistic summary via execution of the following steps: (1) the user formulates a set of linguistic summaries of interest (relevance) using the fuzzy querying add-on, (2) the system retrieves records from the database and calculates the validity of each summary adopted, and (3) the system uses the validity values to select the most appropriate summary.

Dubois and Prade [28] proposed a gradual linguistic summarization in the

form "the more X is F, the more/ the less Y is G" that expresses a progressive change of the degree to which the entity Y satisfies the gradual property G when the degree to which the entity X satisfies the gradual property F is modified.

Inspired by gradual inference rules by Dubois and Prade, Wilbik and Kaymak proposed another type of protoform-based linguistic summary – the gradual summaries – that aimed at capturing the change (increasing/decreasing) within the data points over some time span [102].

Another type of linguistic summary in form of IF-THEN rules whose words used for each antecedent and consequent are user-provided is proposed by Wu et al [105], [106].

In order to discover relations in dynamic data such as time series, new approaches to the linguistic summarization have been introduced by Kacprzyk et al. [51], [55], [56] and Kobayashi et al [63].

6.6 Conclusion

On multiple occasions, the users are not interested in details of data retrieved from data sources, but in their distilled, easily understandable form that explains its essential content. Linguistic summaries and averaging operators are techniques and tools allowing web-based query-answering systems to provide the users with summary answers in natural language.

In this chapter, we propose a method for automatic construction of linguistic summaries of numerical multi-dimensional (multi-feature) data which are answers to the users' questions. Using the method, LingTeQA can select the most suitable summarizer (linguistic term) from a set of linguistic terms provided by the user. It also identifies the most suitable quantifier and can apply constraints on a subset of attributes of the retrieved data.

We also propose a simple Context-based User-defined Weighted Averaging ($CUWA$) operator to determine averages over multi-dimensional data. The data being averaged must satisfy several constraints (context) imposed on their attributes.

The proposed approach mimics the human ability to make conclusions without precise measurements and calculations. Thus, we move a bit closer towards human-centric question answering systems via providing such systems with a limited, at least so far, capability to construct linguistic answers. Suitable linguistic descriptions are selected from the definitions of imprecise concepts provided by the users. To simplify a process of 'entering' definitions of fuzzy sets that reflect the user's perception of the concepts, we utilize a GUI system for drawing such functions.

Chapter 7

Conclusion

7.0.1 Overview of Contributions

A large amount of RDF data stored in connected Knowledge Graphs (KGs) is available for public access. Many of the RDF stores contain millions of triples. In order to take advantage of this data, users have to possess knowledge of KGs' schemas, as well as skills in constructing queries in the RDF quary language (SPARQL). Yet, although such repositories as DBpedia, Wikidata, contain billions of triples they are quite often insufficient for uses' needs, and in addition they may contain conflicting facts. As a result, a process of finding needed information in KGs is not a trivial task for most users.

Many question-answering systems have been developed. However, a significant majority of them are only able to answer simple questions. These systems cannot handle questions that require translating them into complex queries and/or questions that contain linguistic terms. Similarly, they are not capable of fusing data obtained as a result of queries executed over multiple KGs, as well as resolving conflicting information.

As it looks like a step towards a more user-friendly way of retrieving information, we could state that in order to answer some types of questions, especially the ones that require additional processing of information and some post-processing of the obtained results, there is still a need to develop adequate processes and methods. It could be said that users would appreciate asking more comprehensive/in-depth questions, as well as obtaining results in a more condensed form.

The thesis is the result of our five successive years of research aiming at designing and developing a question-answering system that is able to accept questions asked in the user's natural language with user-specific imprecise terms. This system utilizes multiple knowledge repositories, and communicates results in a natural language. Our research results in two journal and five conference papers and a running Web application *LingTeQA* at https://www.lingteqa.site/

LingTeQA allows users to ask questions of higher complexity and to obtain results as they would interact with another human. It is a white-box system containing many distinctive capabilities. In particular, *LingTeQA* is capable of

1) Constructing SPARQL queries for retrieving answers to the users' questions. These queries can be used to explain to the users the answers.

2) Answering complex questions that require queries with multiple triple patterns, functions, and aggregation.

3) Answering questions containing linguistic terms by providing users with a friendly GUI for entering fuzzy sets describing linguistic terms

4) Providing short answers in the format of linguistic summaries for some suitable questions.

5) Extending its template repository by automatically generating templates from user-provided question-query pairs of syntactically unseen questions. As a result, the system evolves over time in the sense that it is able to answer more and more types of questions.

LingTeQA is an efficient system in terms of computational time for the following reasons:

1) Searching a query template for a newly asked is as simpler as looking at a word in a small dictionary;

2) Each query template does not only specify the query structure but also provides clear guidelines for the mapping process indicating which constituent in the asked question must be mapped into which query item.

7.0.2 Future Work

Although *LingTeQA* can answer a wide range of questions well, there are some tasks that can be improved to serve users better.

1) *LingTeQA* can answer syntactically known questions with high accuracy, but it can not answer synonymous questions whose syntactic structures are unknown. It has been shown that paraphrasing can help handle the variation of natural language statements. Although we adopted a paraphrasing approach to extend the system?s template repository, the paraphraser has not been integrated yet into our currently running system (Web app) due to the limited configuration of its host. In the future, we will equip the system with the paraphrasing ability.

2) In order to make a QA system easier to use by more users, it should allow them to ask questions in languages other than English. In principle, the Semantic Web is very well suited for multilingualism, as URIs are language-independent identifiers. In general, natural language processing tools for English are well-developed, and most existing RDF data are in English. However, NLP for non-English languages and RDF data available in other languages are still being developed. Currently, our system can only answer English questions. In the future, it will be extended with the ability to answer questions in languages other than English.

3) The system's GUI will to be improved as well. The user interface should be more attractive and easy to use for a wider range of users. We envision a simple yet effective way of entering shapes of membership functions – using a finger, stylus or mouse – and an easy to manage repository of already defined membership functions.

References

[1] A. Abujabal, R. S. Roy, M. Yahya, and G. Weikum, "Quint: Interpretable question answering over knowledge bases," *EMNLP 2017 - Conference on Empirical Methods in Natural Language Processing: System Demonstrations, Proceedings*, no. ii, pp. 61–66, 2017. DOI: 10.18653/v1/d17-2011.

[2] A. Abujabal, R. Saha Roy, M. Yahya, and G. Weikum, "Never-ending learning for open-domain question answering over knowledge bases," *The Web Conference 2018 - Proceedings of the World Wide Web Conference, WWW 2018*, no. i, pp. 1053–1062, 2018. DOI: 10.1145/3178876.3186004.

[3] P. T. B, G. Maheshwari, and M. Dubey, "LC-QuAD : A Corpus for Complex Question," vol. 1, pp. 210–218, 2017. DOI: 10.1007/978-3-319-68204-4.

[4] C. Bannard and C. Callison-Burch, "Paraphrasing with bilingual parallel corpora," *ACL-05 - 43rd Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference*, pp. 597–604, 2005. DOI: 10.3115/1219840.1219914.

[5] R. Barzilay and K. R. McKeown, "Extracting paraphrases from a parallel corpus," pp. 50–57, 2001. DOI: 10.3115/1073012.1073020.

[6] H. Bast and E. Haussmann, "More accurate question answering on freebase," *International Conference on Information and Knowledge Management, Proceedings*, vol. 19-23-Oct-, pp. 1431–1440, 2015. DOI: 10.1145/2806416.2806472.

[7] J. Berant, A. Chou, R. Frostig, and P. Liang, "Semantic parsing on freebase from question-answer pairs," *EMNLP 2013 - 2013 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*, no. October, pp. 1533–1544, 2013.

[8] T. Berners-Lee, J. Hendler, and O. Lassila, "The Semantic Web," *Scientific american*, vol. 5, no. May, pp. 34–43, 2001.

[9] J. Bleiholder and F. Naumann, "Data Fusion," *ACM Computing Surveys*, vol. 41, no. 1, pp. 1–41, 2009, ISSN: 15577341. DOI: 10.1145/1456650.1456651.

[10] A. Bordes, N. Usunier, S. Chopra, and J. Weston, "Large-scale Simple Question Answering with Memory Networks," 2015. arXiv: 1506.02075. [Online]. Available: http://arxiv.org/abs/1506.02075.

[11] P. Bosc and O. Pivert, "SQLf: A Relational Database Language for Fuzzy Querying," *IEEE Transactions on Fuzzy Systems*, vol. 3, no. 1, pp. 1–17, 1995, ISSN: 19410034. DOI: 10.1109/91.366566.

[12] Q. Cai and A. Yates, "Large-scale semantic parsing via schema matching and lexicon extension," *ACL 2013 - 51st Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference*, vol. 1, pp. 423–433, 2013.

[13] D. Chen, A. Fisch, J. Weston, and A. Bordes, "Reading Wikipedia to answer open-domain questions," *ACL 2017 - 55th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference (Long Papers)*, vol. 1, pp. 1870–1879, 2017. DOI: 10.18653/v1/P17-1171. arXiv: 1704.00051.

[14] L. Chen and C. L. Philip Chen, "Gradient Pre-shaped Fuzzy C-means Algorithm (GradPFCM) for transparent membership function generation," *IEEE International Conference on Fuzzy Systems*, pp. 428–433, 2008, ISSN: 10987584. DOI: 10.1109/FUZZY.2008.4630403.

[15] P. Christmann, R. S. Roy, A. Abujabal, J. Singh, and G. Weikum, "Look before you Hop: Conversational Question Answering over Knowledge Graphs Using Judicious Context Expansion," *arXiv*, pp. 729–738, 2019, ISSN: 23318422.

[16] Z. Dai, L. Li, and W. Xu, "CFO: Conditional Focused neural question answering with large-scale knowledge bases," *54th Annual Meeting of the Association for Computational Linguistics, ACL 2016 - Long Papers*, vol. 2, no. 1, pp. 800–810, 2016. DOI: 10.18653/v1/p16-1076. arXiv: 1606.01994.

[17] D. Damljanovic, M. Agatonovic, and H. Cunningham, "FREyA: An interactive way of querying linked data using natural language," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 7117 LNCS, pp. 125–138, 2012, ISSN: 03029743. DOI: 10.1007/978-3-642-25953-1_11.

[18] M. C. De Marneffe, T. Dozat, N. Silveira, *et al.*, "Universal stanford dependencies: A cross-linguistic typology," *Proceedings of the 9th International Conference on Language Resources and Evaluation, LREC 2014*, pp. 4585–4592, 2014.

[19] D. Diefenbach, A. Both, K. Singh, and P. Maret, "Towards a question answering system over the Semantic Web," *Semantic Web*, vol. 11, no. 3, pp. 421–439, 2020, ISSN: 22104968. DOI: 10.3233/SW-190343. arXiv: 1803.00832.

[20] D. Diefenbach, V. Lopez, K. Singh, and P. Maret, "Core techniques of question answering systems over knowledge bases: a survey," *Knowledge and Information Systems*, vol. 55, no. 3, pp. 529–569, 2018, ISSN: 02193116. DOI: 10.1007/s10115-017-1100-y.

[21] D. Diefenbach, V. Lully, P. H. Migliatti, K. Singh, O. Qawasmeh, and P. Maret, "QAnswer: A question answering prototype bridging the gap between a considerable part of the LOD cloud and end-users," *The Web Conference 2019 - Proceedings of the World Wide Web Conference, WWW 2019*, no. December 2018, pp. 3507–3510, 2019. DOI: 10.1145/3308558.3314124.

[22] D. Diefenbach, K. Singh, and P. Maret, "WDAqua-core0: A question answering component for the research community," *Communications in Computer and Information Science*, vol. 769, pp. 84–89, 2017, ISSN: 18650929. DOI: 10.1007/978-3-319-69146-6_8.

[23] E. Dimitrakis, K. Sgontzos, and Y. Tzitzikas, "A survey on question answering systems over linked data and documents," *Journal of Intelligent Information Systems*, vol. 55, no. 2, pp. 233–259, 2020, ISSN: 15737675. DOI: 10.1007/s10844-019-00584-7.

[24] L. Dong, J. Mallinson, S. Reddy, and M. Lapata, "Learning to paraphrase for question answering," *arXiv*, 2017, ISSN: 23318422.

[25] X. L. Dong, E. Gabrilovich, G. Heitz, *et al.*, "From data fusion to knowledge fusion," *Proceedings of the VLDB Endowment*, vol. 7, no. 10, pp. 881–892, 2014, ISSN: 21508097. DOI: 10.14778/2732951.2732962. arXiv: 1503.00302.

[26] X. L. Dong and D. Srivastava, "Compact explanation of data fusion decisions," *WWW 2013 - Proceedings of the 22nd International Conference on World Wide Web*, pp. 379–389, 2013. DOI: 10.1145/2488388.2488422.

[27] M. Dubey, D. Banerjee, A. Abdelkawi, and J. Lehmann, "LC-QuAD 2.0: A Large Dataset for Complex Question Answering over Wikidata and DBpedia," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 11779 LNCS, pp. 69–78, 2019, ISSN: 16113349. DOI: 10.1007/978-3-030-30796-7_5.

[28] D. Dubois and H. Prade, "Gradual inference rules in approximate reasoning," *Information Sciences*, vol. 61, no. 1-2, pp. 103–122, 1992, ISSN: 00200255. DOI: 10.1016/0020-0255(92)90035-7.

[29] P. A. Duboue and J. Chu-Carroll, "Answering the question you wish they had asked," no. February, pp. 33–36, 2006. DOI: 10.3115/1614049.1614058.

[30] S. K. Dwivedi and V. Singh, "Research and Reviews in Question Answering System," *Procedia Technology*, vol. 10, pp. 417–424, 2013, ISSN: 22120173. DOI: 10.1016/j.protcy.2013.12.378. [Online]. Available: http://dx.doi.org/10.1016/j.protcy.2013.12.378.

[31] A. Fader, L. Zettlemoyer, and O. Etzioni, "Paraphrase-driven learning for open question answering," *ACL 2013 - 51st Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference*, vol. 1, pp. 1608–1618, 2013.

[32] ——, "Open question answering over curated and extracted knowledge bases," *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1156–1165, 2014. DOI: 10.1145/2623330.2623677.

[33] C. Fellbaum, *WordNet : an electronic lexical database*. Cambridge, MA: MIT Press, 1998, p. 423, ISBN: 9780262272551.

[34] S. Ferr and M. Negri, "The QALL-ME Framework : A Specifiable-Domain Multilingual," *Learning*, no. January, 2011.

[35] S. Ferré, "Sparklis: An expressive query builder for SPARQL endpoints with guidance in natural language," *Semantic Web*, vol. 8, no. 3, pp. 405–418, 2017, ISSN: 22104968. DOI: 10.3233/SW-150208.

[36] D. Ferrucci, E. Brown, J. Chu-Carroll, *et al.*, "Building watson: An overview of the deepQA project," *AI Magazine*, vol. 31, no. 3, pp. 59–79, 2011, ISSN: 07384602. DOI: 10.1609/aimag.v31i3.2303.

[37] A. Fujita and P. Isabelle, "Expanding Paraphrase Lexicons by Exploiting Generalities," *ACM Trans. Asian Low-Resour. Lang. Inf. Process*, vol. 17, 2018. DOI: 10.1145/3160488. [Online]. Available: https://doi.org/10.1145/3160488.

[38] J. Ganitkevitch, B. Van Durme, and C. Callison-Burch, "PPDB: The paraphrase database," *NAACL HLT 2013 - 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Proceedings of the Main Conference*, pp. 758–764, 2013.

[39] J. M. Garibaldi and R. I. John, "Choosing membership functions of linguistic terms," *IEEE International Conference on Fuzzy Systems*, vol. 1, pp. 578–583, 2003. DOI: 10.1109/fuzz.2003.1209428.

[40] B. F. Green, A. K. Wolf, C. Chomsky, and K. Laughery, "Baseball: An automatic question-answerer," *Proceedings of the Western Joint Computer Conference: Extending Man's Intellect, IRE-AIEE-ACM 1961*, pp. 219–224, 1961. DOI: 10.1145/1460690.1460714.

[41] S. Hakimov, S. Jebbara, and P. Cimiano, "AMUSE: Multilingual semantic parsing for question answering over linked data," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 10587 LNCS, pp. 329–346, 2017, ISSN: 16113349. DOI: 10.1007/978-3-319-68288-4_20. arXiv: 1802.09296.

[42] Y. Hao, H. Liu, S. He, K. Liu, and J. Zhao, "Pattern-revising Enhanced Simple Question Answering over Knowledge Bases," *(COLING)International Conference on Computational Linguistics*, pp. 3272–3282, 2018.

[43] T. Hasuike, H. Katagiri, H. Tsubaki, and H. Tsuda, "Constructing membership function based on fuzzy shannon entropy and human's interval estimation," *IEEE International Conference on Fuzzy Systems*, no. 22700233, pp. 10–15, 2012, ISSN: 10987584. DOI: 10.1109/FUZZ-IEEE.2012.6251199.

[44] D. M. Herzig, P. Mika, R. Blanco, and T. Tran, "Using On-the-Fly Consolidation," pp. 167–183, 2013.

[45] K. Höffner, S. Walter, E. Marx, R. Usbeck, J. Lehmann, and A. C. Ngonga Ngomo, "Survey on challenges of Question Answering in the Semantic Web," *Semantic Web*, vol. 8, no. 6, pp. 895–920, 2017, ISSN: 22104968. DOI: 10.3233/SW-160247.

[46] M. Honnibal and M. Johnson, "An improved non-monotonic transition system for dependency parsing," *Conference Proceedings - EMNLP 2015: Conference on Empirical Methods in Natural Language Processing*, no. September, pp. 1373–1378, 2015. DOI: 10.18653/v1/d15-1162.

[47] S. Hu, L. Zou, J. X. Yu, H. Wang, and D. Zhao, "Answering natural language questions by subgraph matching over knowledge graphs (extended abstract)," *Proceedings - IEEE 34th International Conference on Data Engineering, ICDE 2018*, vol. 30, no. 5, pp. 1815–1816, 2018. DOI: 10.1109/ICDE.2018.00265.

[48] D. Isern, L. Marin, A. Valls, and A. Moreno, "The unbalanced linguistic Ordered Weighted Averaging operator," *2010 IEEE World Congress on Computational Intelligence, WCCI 2010*, 2010. DOI: 10.1109/FUZZY.2010.5584199.

[49] A Ittycheriah, M Franz, and S Roukos, "IBM's Statistical Question Answering System-TREC-10.," *Trec*, 2001. [Online]. Available: http://trec.nist.gov/pubs/trec10/papers/trec2001.pdf.

[50] S. Iyer, I. Konstas, A. Cheung, J. Krishnamurthy, and L. Zettlemoyer, "Learning a neural semantic parser from user feedback," *ACL 2017 - 55th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference (Long Papers)*, vol. 1, pp. 963–973, 2017. DOI: 10.18653/v1/P17-1089. arXiv: 1704.08760.

[51] J. Kacprzyk and A. Wilbik, "Linguistic summarization of time series using linguistic quantifiers: Augmenting the analysis by a degree of fuzziness," *IEEE International Conference on Fuzzy Systems*, pp. 1146–1153, 2008, ISSN: 10987584. DOI: 10.1109/FUZZY.2008.4630515.

[52] J. Kacprzyk and S. Zadroiny, "Janusz Kacprzyk and Stawomir Zadroiny Systems Research Institute Polish Academy of Sciences," no. 1986, pp. 167–171, 1989.

[53] J. Kacprzyk and S. Zadrożny, "Fquery for Access: Fuzzy Querying for a Windows-Based DBMS," *Fuzziness in Database Management Systems*, pp. 415–433, 1995. DOI: 10.1007/978-3-7908-1897-0_18.

[54] J. Kacprzyk and S. Zadrozny, "On interactive linguistic summarization of databases via a fuzzy-logic-based querying add-on to microsoft access®," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 1625, no. 1995, pp. 462–472, 1999, ISSN: 16113349. DOI: 10.1007/3-540-48774-3_52.

[55] J. Kacprzyk and S. Zadrozny, "Computing with words is an implementable paradigm: Fuzzy queries, linguistic data summaries, and natural-language generation," *IEEE Transactions on Fuzzy Systems*, vol. 18, no. 3, pp. 461–472, 2010, ISSN: 10636706. DOI: 10.1109/TFUZZ.2010.2040480.

[56] J. Kacprzyk and S. Zadrożny, "Linguistic summaries of time series: A powerful and prospective tool for discovering knowledge on time varying processes and systems," *Studies in Fuzziness and Soft Computing*, vol. 325, pp. 65–77, 2015, ISSN: 14349922. DOI: 10.1007/978-3-319-18750-1_5.

[57] J. Kacprzyk and A. Ziolkowski, "Database Queries with Fuzzy Linguistic Quantifiers," vol. 1, no. 3, pp. 474–479, 1986.

[58] R. M. Kathleen, "Paraphrasing questions using given and new information," *Paraphrasing questions using given and new information*, vol. 9, no. 1, pp. 1–10, 1983, ISSN: 1530-9312. DOI: 10.1145/965105.807463.

[59] J. M. Kleinberg, "Authoritative sources in a hyperlinked environment," *The Structure and Dynamics of Networks*, vol. 9781400841, no. 5, pp. 514–542, 2011. DOI: 10.1515/9781400841356.514.

[60] G. J. Klir and B. Yuan, *Fuzzy sets and fuzzy logic : theory and applications*. Upper Saddle River, N.J.: Prentice Hall PTR, 1995, p. 574, ISBN: 0131011715.

[61] J. Ko, E. Nyberg, and L. Si, "A probabilistic graphical model for joint answer ranking in question answering," *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR'07*, pp. 343–350, 2007. DOI: 10.1145/1277741.1277801.

[62] J. Ko, L. Si, E. Nyberg, and T. Mitamura, "Probabilistic models for answer-ranking in multilingual question-answering," *ACM Transactions on Information Systems*, vol. 28, no. 3, 2010, ISSN: 10468188. DOI: 10.1145/1777432.1777439.

[63] M. Kobayashi and I. Kobayashi, "An approach to linguistic summarization based on comparison among multiple time-series data," pp. 1100–1103, 2012. DOI: 10.1109/SCIS-ISIS.2012.6505059.

[64] G. Kobilarov, C. Bizer, S. Auer, and J. Lehmann, "DBpedia - A Linked Data Hub and Data Source for Web and Enterprise Applications," *International World Wide Web Conference*, vol. 18, pp. 1–3, 2009. [Online]. Available: http://www4.wiwiss.fu-berlin.de/bizer/pub/DBpedia-WWW2009-DevTrack-Abstract.pdf.

[65] V. Kumar and S. Joshi, "Incomplete follow-up question resolution using retrieval based sequence to sequence learning," *SIGIR 2017 - Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 705–714, 2017. DOI: 10.1145/3077136.3080801.

[66] J. Kupiec, "MURAX. A robust linguistic approach for question answering using an on-line encyclopedia," *Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Infofmation Retrieval*, pp. 181–190, 1993.

[67] W. Liu, C. Zhang, B. Yu, and Y. Li, "A general multi-source data fusion framework," *ACM International Conference Proceeding Series*, vol. Part F1481, pp. 285–289, 2019. DOI: 10.1145/3318299.3318394.

[68] X. Liu, X. L. Dong, B. C. Ooi, and D. Srivastava, "Online data fusion," *Proceedings of the VLDB Endowment*, vol. 4, no. 11, pp. 932–943, 2011, ISSN: 21508097. DOI: 10.14778/3402707.3402731.

[69] V. Lopez and E. Motta, "Ontology-driven question answering in AquaLog," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 3136, pp. 89–102, 2004, ISSN: 16113349. DOI: 10.1007/978-3-540-27779-8_8.

[70] V. Lopez, V. Uren, M. Sabou, and E. Motta, "Is question answering fit for the semantic web?: A survey," *Semantic Web*, vol. 2, no. 2, pp. 125–155, 2011, ISSN: 15700844. DOI: 10.3233/SW-2011-0041.

[71] D. Lukovnikov, A. Fischer, J. Lehmann, and S. Auer, "Neural network-based question answering over knowledge graphs on word and character level," *26th International World Wide Web Conference, WWW 2017*, pp. 1211–1220, 2017. DOI: 10.1145/3038912.3052675.

[72] S. Lyu, Y. Wang, W. Ouyang, H. Shen, and X. Cheng, "What we vote for? Answer selection from user expertise view in community question answering," *The Web Conference 2019 - Proceedings of the World Wide Web Conference, WWW 2019*, pp. 1198–1209, 2019. DOI: 10.1145/3308558.3313510.

[73] J Mallinson, R Sennrich, and M Lapata, "Paraphrasing Revisited with Neural Machine Translation," 2017.

[74] P. N. Mendes, H. Mühleisen, and C. Bizer, "Sieve: Linked Data quality assessment and fusion," *ACM International Conference Proceeding Series*, pp. 116–123, 2012. DOI: 10.1145/2320765.2320803.

[75] M Merig, "Fuzzy Generalized Aggregation Operators in a Unified Model between the Probability, the Weighted Average and the OWA Operator," pp. 1–7, 2010. DOI: 10.1109/FUZZY.2010.5584795.

[76] J. M. Merigó, "Probabilities in the OWA operator," *Expert Systems with Applications*, vol. 39, no. 13, pp. 11 456–11 467, 2012, ISSN: 09574174. DOI: 10.1016/j.eswa.2012.04.010.

[77] A. Moschitti, "Answer Filtering via Text Categorization in Question Answering Systems," *Proceedings of the International Conference on Tools with Artificial Intelligence*, pp. 241–248, 2003, ISSN: 10636730. DOI: 10.1109/tai.2003.1250197.

[78] S. Narayan, S. Reddy, and S. B. Cohen, "Paraphrase generation from latent-variable PCFGs for semantic parsing," *INLG 2016 - 9th International Natural Language Generation Conference, Proceedings of the Conference*, pp. 153–162, 2016. DOI: 10.18653/v1/w16-6625. arXiv: 1601.06068.

[79] A. Z. Nikolay Radoev and M. Gagnon, "Multilingual Question Answering using Lexico-Syntactic Patterns," vol. 0, 2019.

[80] L. Page, S. Brin, R. Motwani, and T. Winograd, "The PageRank Citation Ranking: Bringing Order to the Web," *Stanford InfoLab*, pp. 111–112, 1999.

[81] S. Park, S. Kwon, B. Kim, S. Han, H. Shim, and G. G. Lee, "Question Answering System using Multiple Information Source and Open Type Answer Merge," pp. 111–115, 2015. DOI: 10.3115/v1/n15-3023.

[82] J. Pasternack and D. Roth, "Making better informed trust decisions with generalized fact-finding," *IJCAI International Joint Conference on Artificial Intelligence*, pp. 2324–2329, 2011, ISSN: 10450823. DOI: 10.5591/978-1-57735-516-8/IJCAI11-387.

[83] E. Pavlick, P. Rastogi, J. Ganitkevitch, B. V. Durme, and C. Callison-Burch, "PPDB 2.0: Better paraphrase ranking, fine-grained entailment relations, word embeddings , and style classification," *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Short Papers), Beijing, China, July 26-31, 2015*, pp. 425–430, 2015.

[84] W. Pedrycz and F. Gomide, *Fuzzy systems engineering: toward human-centric computing*. John Wiley & Sons, 2007, p. 526, ISBN: 9780471788577.

[85] C. Raffel, N. Shazeer, A. Roberts, *et al.*, "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer," *Journal of Machine Learning Research*, vol. 21, pp. 1–67, 2020. arXiv: 1910.10683v3. [Online]. Available: http://jmlr.org/papers/v21/20-074.html..

[86] D. Rasmussen, "Summary SQL - A Fuzzy Tool For Data Mining," vol. 1, no. 98, pp. 49–58, 1997.

[87] E. Riloff and M. Thelen, "A rule-based question answering system for reading comprehension tests," pp. 13–19, 2000. DOI: 10.3115/1117595.1117598.

[88] T. L. Saaty, "WHAT IS THE ANALYTIC HIERARCHY PROCESS ?," 1988.

[89] W. Sebastian, U. Christina, C. Philipp, and B. Daniel, "Evaluation of a Layered Approach to Question Answering over Linked Data, The Semantic Web – ISWC 2012," vol. 7650, no. 00, pp. 362–374, 2012. DOI: 10.1007/978-3-642-35173-0. [Online]. Available: http://link.springer.com/10.1007/978-3-642-35173-0.

[90] S. Shekarpour and S. Auer, ""SINA: semantic interpretation of user queries for question answering on interlinked data" by Saeedeh Shekarpour with Prateek Jain as coordinator," *ACM SIGWEB Newsletter*, no. Summer, pp. 1–1, 2014, ISSN: 1931-1745. DOI: 10.1145/2641730.2641733.

[91] A. Talmor and J. Berant, "The Web as a Knowledge-base for Answering Complex Questions," in *Conference of the North American Chapter of the Association for Computational Linguistics*, long paper at NAACL 2018, 2018, pp. 641–651, ISBN: 1803.06643v1. arXiv: 1803.06643v1. [Online]. Available: https://github.com/.

[92] T. Tanon, M. D. D. Assuncao, E. Caron, *et al.*, "Platypus – A Multilingual Question Answering Platform for Wikidata To cite this version : HAL Id : hal-01730479 Platypus – A Multilingual Question Answering Platform for Wikidata," 2018.

[93] M. Tasnim, D. Collarana, D. Graux, F. Orlandi, and M. E. Vidal, "Summarizing entity temporal evolution in knowledge graphs," *The Web Conference 2019 - Companion of the World Wide Web Conference, WWW 2019*, vol. 07, no. ii, pp. 961–965, 2019. DOI: 10.1145/3308560.3316521.

[94] S. Thoma, A. Thalhammer, A. Harth, and R. Studer, "FUSE: Entity-centric data fusion on linked data," *ACM Transactions on the Web*, vol. 13, no. 2, 2019, ISSN: 1559114X. DOI: 10.1145/3306128.

[95] N. D. To and M. Reformat, "Question-Answering System with Linguistic Terms over RDF Knowledge Graphs," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 2020-Octob, pp. 4236–4243, 2020, ISSN: 21682232. DOI: 10.1109/SMC42975.2020.9282949.

[96] N. D. To, M. Z. Reformat, and R. R. Yager, "Linked open data: Uncertainty in equivalence of properties," *Advances in Intelligent Systems and Computing*, vol. 643, pp. 418–429, 2018, ISSN: 21945357. DOI: 10.1007/978-3-319-66827-7_38.

[97] C. Unger, L. Bühmann, J. Lehmann, A. C. N. Ngomo, D. Gerber, and P. Cimiano, "Template-based question answering over RDF data," *WWW'12 - Proceedings of the 21st Annual Conference on World Wide Web*, pp. 639–648, 2012. DOI: 10.1145/2187836.2187923.

[98] C. Unger, A. Freitas, and P. Cimiano, "An introduction to question answering over linked data," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 8714, pp. 100–140, 2014, ISSN: 16113349. DOI: 10.1007/978-3-319-10587-1_2.

[99] R. Usbeck, R. H. Gusmita, M. Saleem, and A. C. N. Ngomo, "9th challenge on question answering over linked data (QALD-9)," *CEUR Workshop Proceedings*, vol. 2241, pp. 58–64, 2018, ISSN: 16130073.

[100] R. Usbeck, A. C. N. Ngomo, L. Bühmann, and C. Unger, "HAWK - hybrid question answering using linked data," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 9088, pp. 353–368, 2015, ISSN: 16113349. DOI: 10.1007/978-3-319-18818-8_22.

[101] R. Wang and K. Mei, "Analysis of fuzzy membership function generation with unsupervised learning using self-organizing feature map," *Proceedings - 2010 International Conference on Computational Intelligence and Security, CIS 2010*, no. 1, pp. 515–518, 2010. DOI: 10.1109/CIS.2010.118.

[102] A. Wilbik and U. Kaymak, "Gradual Linguistic Summaries," *Communications in Computer and Information Science*, vol. 443 CCIS, no. PART 2, pp. 405–413, 2014, ISSN: 18650929. DOI: 10.1007/978-3-319-08855-6_41.

[103] W. A. Woods, *Semantics and Quantification in Natural Language Question Answering*, C. 1978, vol. 17, pp. 1–87. DOI: 10.1016/S0065-2458(08)60390-3.

[104] D. Wu and J. M. Mendel, "Ordered fuzzy weighted averages and ordered linguistic weighted averages," *2010 IEEE World Congress on Computational Intelligence, WCCI 2010*, 2010. DOI: 10.1109/FUZZY.2010.5584479.

[105] ——, "Linguistic summarization using IF-THEN rules and interval Type-2 fuzzy sets," *IEEE Transactions on Fuzzy Systems*, vol. 19, no. 1, pp. 136–151, 2011, ISSN: 10636706. DOI: 10.1109/TFUZZ.2010.2088128.

[106] D. Wu, J. M. Mendel, and J. Joo, "Linguistic summarization using IF-THEN rules," *2010 IEEE World Congress on Computational Intelligence, WCCI 2010*, 2010. DOI: 10.1109/FUZZY.2010.5584500.

[107] H. Xiaoyan, C. Xiaoming, and L. Kaiying, "A rule-based chinese question answering system for reading comprehension tests," *Proceedings - 3rd International Conference on Intelligent Information Hiding and Multimedia Signal Processing, IIHMSP 2007.*, vol. 2, pp. 325–329, 2007. DOI: 10.1109/IIH-MSP.2007.59.

[108] K. Xu, Y. Feng, S. Huang, and D. Zhao, "Hybrid question answering over knowledge base and free text," *COLING 2016 - 26th International Conference on Computational Linguistics, Proceedings of COLING 2016: Technical Papers*, pp. 2397–2407, 2016.

[109] Y. Xu, P. Martínez-Gómez, Y. Miyao, and R. Goebel, "Paraphrase for Open Question Answering: New Dataset and Methods," pp. 53–61, 2016. DOI: 10.18653/v1/w16-0109.

[110] R. R. Yager, "A new approach to the summarization of data," *Information Sciences*, vol. 28, no. 1, pp. 69–86, 1982, ISSN: 00200255. DOI: 10.1016/0020-0255(82)90033-0.

[111] ——, "Generalized OWA aggregation operators," *Fuzzy Optimization and Decision Making*, vol. 3, no. 1, pp. 93–107, 2004, ISSN: 15684539. DOI: 10.1023/B:FODM.0000013074.68765.97.

[112] R. R. Yager and D. Filev, "Induced ordered weighted averaging operators," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 29, no. 2, pp. 141 –150, 1999. DOI: 10.1109/3477.752789.

[113] R. R. Yager, M. Z. Reformat, and N. D. To, "Drawing on the iPad to input fuzzy sets with an application to linguistic data science," *Information Sciences*, vol. 479, pp. 277–291, 2019, ISSN: 00200255. DOI: 10.1016/j.ins.2018.11.048. [Online]. Available: https://doi.org/10.1016/j.ins.2018.11.048.

[114] R. R. Yager, "On Ordered Weighted Averaging Aggregation Operators in Multicriteria Decisionmaking," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 18, no. 1, pp. 183–190, 1988, ISSN: 21682909. DOI: 10.1109/21.87068.

[115] M. Yahya and K. Berberich, "Robust Question Answering over the Web of Linked Data," *ACM*, p. 10, 2013.

[116] X. Yin, J. Han, and P. S. Yu, "Truth discovery with multiple conflicting information providers on the Web," *IEEE Transactions on Knowledge and Data Engineering*, vol. 20, no. 6, pp. 796–808, 2008, ISSN: 10414347. DOI: 10.1109/TKDE.2007.190745.

[117] L. A. Zadeh, "A prototype-centered approach to adding deduction capability to search engines - The concept of protoform," *2002 1st International IEEE Symposium*, vol. 1, no. September, pp. 2–3, 2002. DOI: 10.1109/IS.2002.1044219.

[118] L. Zadeh, "Fuzzy Sets," *Information and Control*, vol. 8, pp. 338–353, 1965.

[119] S. Zadrozny and J. Kacprzyk, "Fuzzy analytical queries: A new approach to flexible fuzzy queries," *IEEE International Conference on Fuzzy Systems*, vol. 2020-July, 2020, ISSN: 10987584. DOI: 10.1109/FUZZ48607.2020.9177556.

[120] H. Zafar, M. Dubey, J. Lehmann, and E. Demidova, "IQA: Interactive query construction in semantic question answering systems," *Journal of Web Semantics*, vol. 64, p. 100 586, 2020, ISSN: 15708268. DOI: 10.1016/j.websem.2020.100586. arXiv: 2006.11534. [Online]. Available: https://doi.org/10.1016/j.websem.2020.100586.

[121] K. Zhang and J. Zhao, "A Chinese question-answering system with question classification and answer clustering," *Proceedings - 2010 7th International Conference on Fuzzy Systems and Knowledge Discovery, FSKD 2010*, vol. 6, no. Fskd, pp. 2692–2696, 2010. DOI: 10.1109/FSKD.2010.5569607.

[122] X. Zhang, L. Zou, and S. Hu, "An interactive mechanism to improve question answering systems via feedback," *International Conference on Information and Knowledge Management, Proceedings*, pp. 1381–1390, 2019. DOI: 10.1145/3357384.3358059.

[123] W. Zheng, H. Cheng, L. Zou, J. X. Yu, and K. Zhao, "Natural language question/answering: Let users talk with the knowledge graph," *International Conference on Information and Knowledge Management, Proceedings*, vol. Part F1318, pp. 217–226, 2017. DOI: 10.1145/3132847.3132977.

[124] C. Zhu, K. Ren, X. Liu, H. Wang, Y. Tian, and Y. Yu, "A graph traversal based approach to answer non-aggregation questions over DBpedia," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 9544, pp. 219–234, 2016, ISSN: 16113349. DOI: 10.1007/978-3-319-31676-5_16. arXiv: 1510.04780.

[125] E. Zimina, J. Nummenmaa, K. Jarvelin, J. Peltonen, and K. Stefanidis, "MuG-QA: Multilingual Grammatical Question Answering for RDF Data," 2018.

[126] L. Zou, R. Huang, H. Wang, J. X. Yu, W. He, and D. Zhao, "Natural language question answering over RDF - A graph data driven approach," *Proceedings of the ACM SIGMOD International Conference on Management of Data*, no. February 2015, pp. 313–324, 2014, ISSN: 07308078. DOI: 10.1145/2588555.2610525.