



**BRNO UNIVERSITY OF TECHNOLOGY**

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

**FACULTY OF INFORMATION TECHNOLOGY**

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

**DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA**

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

**QUESTION ANSWERING OVER STRUCTURED DATA**

ODPOVÍDÁNÍ NA OTÁZKY NAD STRUKTUROVANÝMI DATY

**MASTER'S THESIS**

DIPLOMOVÁ PRÁCE

**AUTHOR**

AUTOR PRÁCE

**MARK BIRGER**

**SUPERVISOR**

VEDOUCÍ PRÁCE

**Doc. RNDr. PAVEL SMRŽ, Ph.D.**

**BRNO 2017**

**Brno University of Technology - Faculty of Information Technology**

Department of Computer Graphics and Multimedia

Academic year 2016/2017

**Master's Thesis Specification**

For: **Birger Mark, Bc.**  
Branch of study: Intelligent Systems  
Title: **Question Answering over Structured Data**  
Category: Artificial Intelligence

Instructions for project work:

1. Get familiar with the design and implementation of question answering systems, focusing on communication in (restricted) natural language.
2. Design and implement a system, which will translate natural language queries for structured knowledge bases such as DBpedia or WikiData and will answer them based on the data contained.
3. Collect data for evaluation and assess the quality of results using standard metrics.
4. Prepare a poster presenting work objectives, the approach followed and its results.

Basic references:

- QALD - a series of evaluation campaigns on question answering over linked data - <http://qald.sebastianwalter.org>

Requirements for the semestral defense:

- Functional prototype

Detailed formal specifications can be found at <http://www.fit.vutbr.cz/info/szz/>

The Master's Thesis must define its purpose, describe a current state of the art, introduce the theoretical and technical background relevant to the problems solved, and specify what parts have been used from earlier projects or have been taken over from other sources.

Each student will hand-in printed as well as electronic versions of the technical report, an electronic version of the complete program documentation, program source files, and a functional hardware prototype sample if desired. The information in electronic form will be stored on a standard non-rewritable medium (CD-R, DVD-R, etc.) in formats common at the FIT. In order to allow regular handling, the medium will be securely attached to the printed report.

Supervisor: **Smrř Pavel, doc. RNDr., Ph.D.**, DCGM FIT BUT

Beginning of work: November 1, 2016

Date of delivery: May 24, 2017

**VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ**  
Fakulta informačních technologií  
Ústav počítačové grafiky a multimédií  
612 66 Brno, Štěrbohova 2



Jan Černocký

*Associate Professor and Head of Department*

## Abstract

This thesis deals with question answering over structured data. In knowledge databases, a structured data is usually represented by graphs. However, to satisfy information needs using natural language interfaces the system is required to hide the underlying schema from users. A question answering system with a schema-agnostic graph-based approach was developed as a part of this work. In contrast to traditional question answering systems that rely on deep linguistic analysis and statistical methods, the developed system explores provided graph to yield and reuse semantic connection for a known question-answer pair. Lack of large domain-specific structured data made us perform evaluation with the help of prominent open linked datasets such as Wikidata and DBpedia. Quality of separate answering stages and the approach in general was evaluated using adapted evaluation dataset and standard metrics.

## Abstrakt

Tato práce se zabývá problematikou odpovídání na otázky nad strukturovanými daty. Ve většině případů jsou strukturovaná data reprezentována pomocí propojených grafů, avšak ukrytí koncové struktury dat je podstatné pro využití podobných systémů jako součástí rozhraní s přirozeným jazykem. Odpovídající systém byl navržen a vyvíjen v rámci této práce. V porovnání s tradičními odpovídajícími systémy, které jsou založené na lingvistické analýze nebo statistických metodách, náš systém zkoumá poskytnutý graf a ve výsledku generuje sémantické vazby na základě vstupních párů otázka-odpověď. Vyvíjený systém je nezávislý na struktuře dat, ale pro účely vyhodnocení jsme využili soubor dat z Wikidata a DBpedia. Kvalita výsledného systému a zkoumaného přístupu byla vyhodnocena s využitím připraveného datasetu a standardních metrik.

## Keywords

question answering system, structured data, linked data, natural language processing, question answering over linked data, natural interface, graph-based approach, schema-agnostic

## Klíčová slova

odpovídání nad strukturovanými daty, odpovídající systém, strukturovaná data, propojena data, zpracování přirozeného jazyků, přirozeně rozhraní

## Reference

BIRGER, Mark. *Question Answering over Structured Data*. Brno, 2017. Master's thesis. Brno University of Technology, Faculty of Information Technology. Supervisor Doc. RNDr. Pavel Smrž, Ph.D.

# Question Answering over Structured Data

## Declaration

Hereby I declare that this master's thesis was prepared as an original author's work under the supervision of Mr. Smrž. All the relevant information sources, which were used during preparation of this thesis, are properly cited and included in the list of references.

.....  
Mark Birger  
May 24, 2017

# Contents

<b>1</b>	<b>Introduction</b>	<b>4</b>
1.1	Motivation . . . . .	4
1.2	Goals . . . . .	5
1.3	Outline . . . . .	6
<b>2</b>	<b>State of the Art</b>	<b>7</b>
2.1	Question Answering . . . . .	7
2.2	Question answering systems . . . . .	8
2.3	Structured data . . . . .	9
2.4	Natural language processing . . . . .	13
2.5	Question answering over linked data . . . . .	16
2.6	Relevant applications and technologies . . . . .	18
2.7	Trends and open topics . . . . .	19
<b>3</b>	<b>Design of the system</b>	<b>21</b>
3.1	Requirements . . . . .	21
3.2	Processing pipeline . . . . .	22
3.3	Analysis of features . . . . .	23
3.4	Algorithm description . . . . .	24
3.5	Processing examples . . . . .	24
<b>4</b>	<b>Implementation</b>	<b>35</b>
4.1	Architecture . . . . .	35
4.2	Modules . . . . .	35
4.3	Linked data integration . . . . .	36
4.4	Third-party components . . . . .	37
<b>5</b>	<b>Evaluation</b>	<b>39</b>
5.1	Datasets . . . . .	39
5.2	Evaluation measures . . . . .	41
5.3	Comparison with existing solutions . . . . .	47
<b>6</b>	<b>Conclusions and potential extensions</b>	<b>49</b>
	<b>Bibliography</b>	<b>51</b>
<b>A</b>	<b>CD Content</b>	<b>57</b>
<b>B</b>	<b>Manual</b>	<b>58</b>

B.1	Virtual environment installation . . . . .	58
B.2	External dependencies . . . . .	59
B.3	Getting started . . . . .	59
<b>C</b>	<b>Evaluation dataset</b>	<b>60</b>
<b>D</b>	<b>Poster</b>	<b>63</b>

# List of Figures

2.1	Common components of a question answering system. . . . .	17
3.1	Designed processing pipeline of the question answering system. . . . .	22
3.2	Selecting noun subtrees from the syntax tree. . . . .	25
3.3	Structural and synonymic permutations of the subtree. . . . .	27
3.4	Answering with a reference scheme. . . . .	29
3.5	Extension of the question-answer pairs database scheme. . . . .	30
3.6	Answering an unknown question over linked data graph. . . . .	32
4.1	Architecture of the developed question answering system. . . . .	36
5.1	Components of the final evaluation dataset. . . . .	41
5.2	Solutions and semantic paths characteristics. . . . .	43
5.3	Processing time of the processing pipeline's stages. . . . .	45
5.4	Example of a tree matching. . . . .	46
D.1	Research illustration. . . . .	63

# Chapter 1

## Introduction

### 1.1 Motivation

In recent years, the usage of Natural Language Processing (NLP) for natural language interfaces towards an effective Human–Computer Interaction (HCI) has received much attention [59]. In last few years we see how dialog interfaces, both spoken and textual are coming into daily life through available commercial products. Personal assistants like Siri, Google Assistant and Cortana are examples of multi-purpose question answering systems which provide to the end-user natural language answers based on structured information. Monthly active user audience of messaging apps exceeds social network users audiences and this trend give a rise to a new market of chatbots. Conversational interfaces become more popular with the progress of natural language processing techniques [16].

Companies of all sizes have an interest to apply such technologies in their services because it allows them to optimize support expenses in case of existing users and opens the way to new communication channels of marketing among potential customers. A development of a conversational agents over popular messaging apps is cheaper than the development of a mobile application, but provides almost the same features and potential benefits to a company. Providing a customer service of a permanent quality is possible with Natural Language Processing technologies. Automated solutions are much more scalable in comparison with any human powered support/sales service. Billing services in messaging apps create a new market.

While marketing new products or providing a support, an employee uses prepared semi-structured data of the company (e.g. deals information, conversation scripts). Conceptual data description formats, such as Resource Description Framework (RDF), are efficient tool to manage that. Most of semi-structured data can be converted into unified knowledge representations. Actual principles and techniques, used in question answering systems over knowledge bases, are applicable to a companies' semi-structured data. Nowadays, application of a modern natural language techniques in this area requires certain customer's skills and domain-specific knowledge. Key goal is to abstract users from the representation of the dataset, allowing to edit and to improve automated solutions with a low entry barrier.

The exponential growth of the World Wide Web has transformed it into a knowledge ecosystem in which highly variative information is linked in an extremely complex and arbitrary manner [35]. It is important to note that the Web is increasingly understood as a global information space consisting not only of linked documents, but also of Linked Data [19]. The rapid increase in massive information storage and the popularity of using the Web allow researchers to store data and make them available to the public. However, the



exploration of this large amount of data makes finding information a complex and time-consuming task [21]. Knowledge bases play an important role in enhancing of the Web and enterprise search intelligence, as well as in supporting information integration [20]. In dealing with Semantic Web databases that can be distributed among multiple computers with different database management systems on remote sites, a central problem faced by users is the query formulation in terms communicable to the system [33]. Natural language is a powerful tool of human-computer interaction, but any data processing systems require knowledge about the words meaning [26].

The key challenge for commercial conversational agents and question answering engines is to transform natural language question into structured query, interpretable by the knowledge base management system. Over the past years, a range of approaches have been developed to address this challenge, showing significant advances towards answering natural language questions with an accent to large, heterogeneous sets of structured data [58]. Nowadays, available structured data is distributed among collection of interconnected datasets, partly available only in textual form. Datasets inconsistent structure affects question answering systems design. New approaches in this area try to solve mentioned issues. Current evaluation campaigns, such as Question Answering over Linked Data (QALD) campaigns, help developers and researchers to evaluate the quality of developed systems.

## 1.2 Goals

The aim is to design and implement schema-agnostic question answering system. Our system will be based on the approach of natural language queries answering over knowledge graphs using a reference question, therefore the evaluation of the approach is necessary. The system should focus on the structured data with the high measure of granularity represented at the linked graph.

These objectives were selected:

1. Create a general domain question answering system, that:
  - (a) uses schema-agnostic approach;
  - (b) is dataset independent.
2. Create an extensible solution:
  - (a) that is independent of a language's linguistic structures;
  - (b) abstracts data consumers from the dataset representation;
  - (c) extensible using natural language.
3. Modular architecture of the system.
4. Evaluate the proposed approach:
  - (a) over existing datasets;
  - (b) using standard or adapted metrics.

Traditional approaches are based on linguistic structures or formal grammars. We will focus on the graph explorations to escape the dependence of a linguistic structure. Our

system should provide sufficient level of abstraction and hide data representation in most cases. The system should be extensible by users (at least ones with a specific role assigned). The proposed approach should link natural language relations with corresponding semantic relations if such relations are available within a knowledge base. Modularity of question answering system is required to make developed components reusable.

Evaluation on the existing dataset, while meeting the lack of large-scale companies' data, is possible using prominent open knowledge bases (Wikidata and DBpedia). Our custom approach requires an adaptation of evaluation datasets, hence the objective is to create the adapted dataset for evaluation. Evaluation using standard metrics is an essential goal because the approach is valuable in a comparison with related systems. User evaluation is a necessary part of this work as well since the question answering process is a subject of human-computer interaction and quality of the result depends on the user experience.

### 1.3 Outline

The rest of this thesis is organized as follows:

**Chapter 2: State of the art.** The survey of question answering systems over linked data is provided in this chapter. Actual principles of question answering systems development with an accent on the natural language processing and its relation to the linked data as a form of knowledge representation were researched. Key challenges and trends of question answering systems design constitute significant part of this chapter.

**Chapter 3: Design of the system.** Set of requirements to the proposed system was defined as a result of the survey. The definition is followed by appropriate methods definition, high-level architecture of the system, and an example of question processing.

**Chapter 4: Implementation.** This chapter explains how the question answering system was developed, provides technical details and summarizes technologies used in the system.

**Chapter 5: Evaluation.** The chapter reveals difficulties of question answering systems evaluation and presents evaluation using adapted datasets. The question answering system was assessed using quality measures in the chapter. Technical comparison with existing question answering systems over linked data is provided at the end of the chapter.

**Chapter 6: Conclusions and potential extensions.** The chapter summarizes achieved results and research conclusions, and describes potential extensions, which can be applied to the system in the future.

# Chapter 2

## State of the Art

### 2.1 Question Answering

Question Answering (QA) is an application area of computer science which attempts to build software systems that can provide accurate useful answers to questions posed by human users in natural language (e.g., English) [29].

As for human-computer interaction, natural language is the best information access mechanism for humans. Therefore, Question Answering Systems (QAS) have special significance and advantages over search engines and are considered as a goal of semantic Web research for user's information needs [56].

Question answering is the process performed by informational system with a natural language interface and can be classified to several categories. Spoken question answering systems in comparison to textual question answering systems have an extra pre-processing step of utterance recognition and extra post-processing step of speech synthesis. Also question answering process is always associated with an information represented by knowledge bases (more at section 2.3). Quality of answers is represented by correctness and completeness metrics and highly depends on a knowledge base quality [57].

#### Questions classification

Natural language question is the input of the process. This section provides classification and examples of different question types.

*Factoid questions* are those asked about Named Entities (NE) or nouns, using for example these words: When, Where, How much/many, Who, and What. These questions ask about date/time, place, person, and organization [21]. Factoid questions also can be separated to multiple classes: predicative questions, list questions, Yes/No questions [57]. There are examples of Factoid questions:

- Who was the first president of the USA?
- What is globalization?
- How far is the New York marathon?
- When was London founded?
- Where was born Mother Teresa?

*Definitions questions* require definition of a term or a concept in result [21]. Answering of this kind of questions is performed by hybrid approaches which uses structured and textual information at the same time. Constructing definition from a structured data is an opposite way of handling this kind of questions. Example:

- Who was George Washington?

*Explanations questions* category is the most complex type of questions, which can require derivation from the provided knowledge. This type of questions tends to have less regular answers structure in comparison to Factoid questions. Examples of these questions are:

- What is the difference between Alzheimer's and dementia?
- What is the connection between history and government?
- Why did they give this name to Gulliver?
- How do I make a cheesecake?
- What does society think of global warming?

### **Answer representation**

Question answering systems are not required, but it is desirable to present answers the same natural way as a user question. Especially for question answering systems over linked data, which purpose is to hide formal semantic queries and structure of the data set from end-user, it is necessary to construct natural language answer possibly enriched by media data [57].

## **2.2 Question answering systems**

A Question Answering System (QAS) is a software system that provides exact answers to natural language questions for some range of topics. The answer of QA system may be supplemented with an additional information, including a clarification or dialog explaining why the provided answer is correct [29]. A Question Answering System aims at giving precise answers to users' questions introduced in natural language [21].

### **Question answering systems vs. dialog systems**

Question answering systems and dialog systems belong to the one field of research. Question answering can be integrated into an existing dialog system as a part of a conversation flow. On the other side, dialog elements with conversation flow can be integrated into a question answering system. Both of them are able to generate natural language answers and the system can be classified as question answering if:

- it has an underlying knowledge base;
- the main purpose is to provide information to the user;
- the context is beneficial, but not compulsory part of the process.

On the other side, the question answering system can be classified as a dialog system if:

- dialog flow is an interface to the underlying layer;
- the primary (but not exclusive) purpose is to control an underlying system;
- a conversation flow generates context monitored by the system.

### Question answering systems vs. search engines

In comparison to regular search engines, question answering systems allow the user to ask a question in natural language and return a correct and exact answer to his question instead of a set of relevant documents [21]. We should note that modern search engines attempt to include semantic answers, mostly based on structured data represented by knowledge graph into search results. That sort of semantic technologies intends to improve visibility and clarity of search results. Google, Bing, Baidu and Yahoo search engines contain semantic technologies based on structured data [45]:

- Knowledge Graph by Google;
- Satori by Microsoft (as a part of Bing);
- knowledge graph by Yahoo;
- Baidu Knowledge Graph.

The above mentioned systems are combining internal graph data parsed during the web crawling and external data sets (for example, Freebase knowledge base was used as a basis for the Google’s Knowledge graph) [50].

### Background

The goal of QA Systems, as defined in [49], is to allow users to ask questions in Natural Language (NL), using their own terminology, and receive a concise answer. The process starts by linguistic analysis (dependency graphs creation using a syntactic parser with a step of a named entities recognition). The next step is to classify the question according to defined questions categories. The query is generated using system-specific approach. An external ontology can be used for matching items generated in the process. Finally, when the query is generated and performed over the Linked Data, the system generates an answer to the user’s question [21].

Question answering systems can be built on top of structured knowledge bases (linked data) or on top of textual data sets, as it has been already done in search engines. Some question answering systems combine both principles (for example IBM’s Watson DeepQA contains a semi-structured knowledge base).

This thesis is focused on question answering over linked data, thus, the architecture of question answering system and key challenges will be reviewed in the section 2.5.

## 2.3 Structured data

Linked Data is a structured data in the terminology of knowledge bases. Linked Data employs the Resource Description Framework (RDF) and the HyperText Transfer Protocol (HTTP). Published structured data on the Web are connected between different data

sources. It allowing data from the one data source to be linked to data in another data source. The principles of Linked Data were first outlined by Berners-Lee in 2006 [14]. This publication includes comprehensive guidance upon which data publishers have begun to realize the Web of Data. Later the guidance has been extended by technical documents that capture best practices from the Linked Data community [19].

Linked data is an example of structured data. As long as we consider the latent linguistic structure of human languages, almost no data are truly “unstructured” [42]. Graph is commonly used for the representation of entities and relations between them.

## Information retrieval

As for academic field of study, information retrieval might be defined as finding material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within large collections (usually stored on computers) [42].

## RDF data triples

Resource Description Framework (RDF) is a standardized model for data interchange on the Web. RDF has features that making possible a data merging even if the underlying schemes differ. Also, RDF supports the evolution of schemes over time without requiring all the data to be changed. RDF uses Universal Resource Identifiers (URIs) to name the relationship between things as well as the two ends of the link (this is usually referred to as a “triple”). Usage of this simple model allows structured and semi-structured data to be mixed, exposed, and shared across different applications. This linking structure forms a directed, labeled graph, where the edges represent the named link between two resources, represented by the graph nodes. The graph view is the possible mental model for RDF and is often used in easy-to-understand visual explanations [47].

Here is provided an example<sup>1</sup> of the RDF triple for the Wikidata entity `wd:Q7874` (American Bobtail, the cat breed):

Listing 2.1: RDF data example.

```
1 <http://www.wikidata.org/entity/Q7874>
2 <http://www.wikidata.org/prop/direct/P31>
3 <http://www.wikidata.org/entity/Q43577> .
4
5 <http://www.wikidata.org/entity/Q43577>
6 <http://www.w3.org/2000/01/rdf-schema#label>
7 "cat breed"@en .
8
9 <http://www.wikidata.org/entity/Q43577>
10 <http://www.w3.org/2004/02/skos/core#prefLabel>
11 "cat breed"@en .
12
13 <http://www.wikidata.org/entity/Q43577>
14 <http://schema.org/name>
15 "cat breed"@en .
16
```

---

<sup>1</sup>Source: <http://www.wikidata.org/wiki/Special:EntityData/Q7874.nt>

```

17 <http://www.wikidata.org/entity/Q7874>
18 <http://www.wikidata.org/prop/P31>
19 <http://www.wikidata.org/entity/statement/Q7874-01F55EE3F372> .
20
21 <http://www.wikidata.org/entity/statement/Q7874-01F55EE3F372>
22 <http://wikiba.se/ontology-beta#rank>
23 <http://wikiba.se/ontology-beta#NormalRank> .
24
25 <http://www.wikidata.org/entity/statement/Q7874-01F55EE3F372>
26 <http://www.wikidata.org/prop/statement/P31>
27 <http://www.wikidata.org/entity/Q43577> .

```

---

RDF data contains semantic triples consisted of subject, predicate and object. The first triple represents a link between subject `wd:Q7874` (American Bobtail) via the predicate `wdt:P31` (instance of) to the object `wd:Q43577` (cat breed). RDF data provided by the Wikidata endpoint for the `wd:Q7874` includes 4597 triples. The request to the endpoint provides extra properties of the connected entities. Furthermore, RDF store contain service ontology subjects such as:

`http://www.wikidata.org/entity/statement/Q7874-01F55EE3F372.`

## SPARQL

As noted before, RDF is a directed, labeled graph data format for representing information in the Web. The specification [46] defines the syntax and semantics of the SPARQL Protocol and RDF Query Language (SPARQL) query language for RDF. SPARQL can be used to express queries across various data sources, whether the data is stored natively as RDF or viewed as RDF via middleware (for example, Wikidata stores knowledges in JavaScript Object Notation (JSON) blobs and generates RDF triples for the current version of the data). SPARQL contains capabilities for querying graph patterns along with their conjunctions and disjunctions. The result of a SPARQL query is a set of entities or an RDF graph. SPARQL is SQL-like querying language. Below you can see an example of the SPARQL query with Wikidata-specific items.

Listing 2.2: SPARQL query example.

---

```

1 SELECT ?breed ?breedLabel ?pic
2 WHERE
3 {
4     ?breed wdt:P31 wd:Q43577 .
5     OPTIONAL {
6         ?breed wdt:P18 ?pic
7     }
8     SERVICE wikibase:label { bd:serviceParam wikibase:language "en" }
9 }

```

---

At the example `?breed` entities were selected as a set of entities with a connection via the property `wdt:P31` (instance of) to the instance `wd:Q43577` (cat breed). Selected breeds were extended with images via the property `wdt:P18` (image of).

## Open knowledge databases

**DBpedia** is a crowdsourced community effort to extract structured information from Wikipedia and make this information available on the Web. DBpedia allows users to ask sophisticated queries against Wikipedia, and to link the different data sets on the Web to Wikipedia data [25]. The aim is to make it easier for the huge amount of information in Wikipedia to be used in some novel ways. Furthermore, DBpedia is potentially may serve new mechanisms for navigating, linking, and improving the Wikipedia itself [37]. Virtuoso SPARQL Query Editor is a publicly available DBpedia SPARQL endpoint <sup>2</sup>.

**Wikidata** is a free and open knowledge base, it can be read and edited by both humans and machines using the MediaWiki engine. Wikidata is the community-created knowledge base of Wikipedia, and the central data management platform for Wikipedia and most of Wikimedia Foundation projects [61]. Wikidata also provides support to many other sites and services beyond just Wikimedia projects. The content of Wikidata is available under a free license, exported using standard formats, and can be interlinked to other open data sets on the linked data web. Wikidata was filled up with Freebase knowledge base after acquiring Metaweb Technologies by Google and shutdown of Freebase. Since its public launch in late 2012, the site has gathered data on more than 24 million entities, including over 130 million statements, and over 138 million labels and descriptions in more than 350 languages [27]. This is the work over 16 thousand users who have actively contributed so far (more than one edit per month). Their efforts continue to make Wikidata more and more comprehensive and accurate [62]. Wikidata SPARQL endpoint is available online <sup>3</sup>.

**YAGO** (Yet Another Great Ontology) is a lightweight and extensible ontology with high coverage and quality. YAGO builds on entities and relations, is includes the Is-A hierarchy as well as subclass relations between entities (such as hasWonPrize). The facts have been automatically extracted from Wikipedia and unified with WordNet, using a carefully designed combination of rule-based and heuristic methods. The resulting knowledge base is a major step beyond WordNet: in quality by adding knowledge about individuals like persons, organizations, products, etc. with their semantic relationships and in quantity by increasing the number of facts. Empirical evaluation of fact correctness shows an accuracy of about 95% [53]. YAGO2 introduced an enhanced data representation, including time and location as first-class citizens. The wealth of temporospatial information in YAGO2 can be explored either graphically or through a special query language [34]. YAGO3 is extension of the YAGO knowledge base that combines the information from the Wikipedia in multiple languages. Special technique fuses the multilingual information with the English WordNet to build one consistent knowledge base [41].

## Proprietary knowledge databases

**Knowledge Graph** is knowledge base used by Google, which helps users to discover new information in search results. It currently contains more than 500 million objects, as well as more than 3.5 billion facts about and relationships between these different objects. The Knowledge Graph Search API allows developers to find entities in the Google Knowledge Graph [50].

---

<sup>2</sup><https://dbpedia.org/sparql/>

<sup>3</sup><https://query.wikidata.org/>



## Unified data set structure

Knowledge bases with structured data, open ones or proprietary, contains diverse information from different areas, structured by various schemes and object classifications of various accuracy. The common thing between all of them is graph representation, which is easily convertible to a set of triples. Data set independent question answering system, which allows interlinking between objects from different data sets should limit the plenty of features available from certain knowledge bases.

## Implementation

**Apache UIMA** (Unstructured Information Management Application) is a software systems suite, that analyze large volumes of unstructured information in order to discover relevant knowledge [2]. An example UIM application might ingest plain text and identify entities, such as persons, places, organizations or relations. UIMA enables applications to be decomposed into components, further each component implements interfaces defined by the framework and provides self-describing metadata via XML descriptor files. The framework manages these components and the data flow between them. UIMA additionally provides capabilities to wrap components as network services, and can scale to very large volumes by replicating processing pipelines over a cluster of networked nodes.

The **Apache Lucene**<sup>TM</sup> project develops open source information retrieval software library, including:

- *Lucene Core*, that provides Java-based indexing and search technology, as well as spellchecking, hit highlighting and advanced analysis/tokenization capabilities [1].
- *Solr*<sup>TM</sup> is an open source enterprise search platform, written in Java, with XML/HTTP and JSON/Python/Ruby APIs. It uses the Lucene search library at its core for full-text indexing and search. Its major features include full-text search, hit highlighting, faceted search, real-time indexing, dynamic clustering, database integration, NoSQL features and rich document handling. Providing distributed search and index replication, Solr is designed for scalability and fault tolerance.

Numerous **RDF JS libraries** are available:

- *rdfstore-js* is JavaScript (JS) RDF store with SPARQL support is a pure Javascript implementation of an RDF graph store with support for the SPARQL query and data manipulation language [7].
- *rdflib.js* is Linked Data API, RDF library for browsers and Node.js, that allowing to read and write RDF/XML, Turtle and N3 [4].
- *node-rdf* are ECMAScript libraries for handling RDF data, designed for Node.js. An important point is that the system implement RDF datatypes with Javascript types and provide related APIs and in-memory utilities[3].

## 2.4 Natural language processing

Natural Language Processing (NLP) techniques can be used to convert the user request from NL into SPARQL [21]. Text processing techniques such as a stemming and lemmatization are used in question answering systems at most stages of input phrase processing.

Lemmatization is widely used at mapping (with the data) step of question answering systems because dictionary form is used while searching in labels indexes of a dataset.

## Common tools

This section reviews prominent tools for natural language processing.

**NLTK** (Natural Language Toolkit) is a leading platform for building Python software to work with human language data [17]. It provides easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning, wrappers for popular NLP libraries [6].

**Stanford CoreNLP** Stanford CoreNLP provides a set of natural language analysis tools [43]. It can give the base forms of words, their parts of speech, whether they are names of companies, people, etc., normalize dates, times, and numeric quantities, mark up the structure of sentences in terms of phrases and word dependencies, indicate which noun phrases refer to the same entities, which is extremely useful for the context monitoring, indicate sentiment, extract particular or open-class relations between entity mentions, get quotes people said, etc [9].

## Parse tree

Syntactic parsing of natural language sentences is a central task in natural language processing (NLP) because of its importance in mediating between linguistic expression and meaning [52].

The **Link Grammar Parser** is a syntactic parser of English, based on link grammar, an original theory of English syntax. Given a sentence, the system assigns to it a syntactic structure, which consists of a set of labeled links connecting pairs of words. The parser also produces a representation of a sentence (showing noun phrases, verb phrases, etc.). The system is written in generic C code and runs on any platform with a C compiler. There is an application program interface (API) to make it easy to incorporate the parser into other applications. Davy Temperley, Daniel Sleator and John Lafferty developed a formal grammatical system called a link grammar, showed how English grammar can be encoded in such a system, including algorithms for efficiently parsing with a link grammar [18]. Although the expressive power of link grammars is equivalent to context free grammars, encoding natural language grammars appears to be much easier with the new system. The performance of this system both in the breadth of English phenomena that it captures and in the computational resources used indicates that the approach may have practical uses [51].

**spaCy** is a library for advanced natural language processing in Python and Cython that was designed to be used in real products. spaCy currently supports English, German and French [5]. Tokenization is based on Penn Treebank script and regular expressions with various updates to account for unicode characters. spaCy tagger uses greedy decoding with the averaged perceptron, Brown cluster features and case normalization features. The parser uses the algorithm of shift-reduce dependency parsing described in the CoNLL 2013 paper (Honnibal, Goldberg and Johnson 2013) with Brown cluster features and redesigned feature set. Other improvements were obtained using Goldberg and Nivre (2012) dynamic oracle with the improved cost-sensitive update.

## Named entities recognition

The goal of a Named Entity (NE) extractor (as a part of the NLP tools family) is to extract named entities. The first definition of NE was coined by Grishman et al. as an information unit such as the name of a person or an organization, a location, a brand, a product, a numeric expression including time, date, money and percent found in a sentence [48].

**NERD** (Named Entity Recognition and Disambiguation) is a web application plugged on top of various named entities extractors. It allows the user to analyze any textual resource published on the web and accessible with a URI, and to extract from the text the named entities detected, typed and disambiguated by five Named Entity Recognition (NER) APIs. It provides a user interface for assessing the performance of each of those five tools according to the pattern (NE,type,URI). All user interactions are collected and stored in a database. The framework can finally generate analysis reports and comparison of tools using the NERD ontology [48].

**TextBlob** is a Python (both 2 and 3) library for processing textual data. It provides a simple API for diving into common natural language processing (NLP) tasks such as part-of-speech tagging, noun phrase extraction, sentiment analysis, classification, translation, and more [10]. TextBlob API contains noun phrase extraction methods, which are even more useful than named entities at the mapping with the linked data step.

**DBpedia Spotlight** is an open source project developing a system for automatic annotation of DBpedia entities in natural language text. It provides programmatic interfaces for phrase spotting (recognition of phrases to be annotated) and disambiguation (entity linking) as well as various output formats (XML, JSON, RDF, etc.) in a REST-based (Representational State Transfer) web service. The standard disambiguation algorithm is based upon cosine similarities and a modification of TF-IDF weights (using Apache Lucene). In comparison to other NERs tools, this one is deeply integrated with the knowledge base and provides an annotated result with a mapping to DBpedia entities [23].

## Synonyms and semantic relations

**WordNet** is a large lexical database of English. Nouns, verbs, adjectives and adverbs are grouped into sets of cognitive synonyms (synsets), each expressing a distinct concept. Synsets are interlinked by means of conceptual-semantic and lexical relations. The resulting network of meaningfully related words and concepts can be navigated with the browser. WordNet is also freely and publicly available for download. WordNet's structure makes it a useful tool for computational linguistics and natural language processing. WordNet superficially resembles a thesaurus, in that it groups words together based on their meanings. However, there are some important distinctions. First, WordNet interlinks not just word forms — strings of letters, but specific senses of words. As a result, words that are found in a close proximity to one another in the network are semantically disambiguated. Second, WordNet labels the semantic relations among words, whereas the groupings of words in a thesaurus does not follow any explicit pattern other than meaning similarity [11]. WordNet provides a more effective combination of traditional lexicographic information and modern computing. In summary, WordNet is an online lexical database designed for use under program control [44].

## 2.5 Question answering over linked data

Current trends confirm that question answering systems are moving to the usage of linked data [21]. In comparison to textual-based, question answering systems based on linked data deeply track semantic relations inside of a question. On the other side, these systems are facing issues with representation of answers and with a construction of natural language results. Besides, the structure of the required information affects the accuracy of those systems. Question answering systems are effective tools to interact with structured knowledge bases [21].

### Key challenges

Traditional approach for question answering method is constructing a query (SPARQL) based on an input question phrase. Most of key challenges are derived from distinctions in expression compatibility between a natural language question and a machine-readable query.

**Variety of semantic terms** complicating mapping possibilities of question answering systems. It is required to use synonyms of word or collocation from the input phrase and properly stem word before searching in knowledge base. Properties or object labels in linked databases are not always provided, but it's commonly used way to map phrase elements to linked data elements [21].

Who was the wife of Zeus?

(connected via P26 property spouse to Hera, Q38012)

**Granularity of language** is another problem, which should be handled by question answering systems. In some cases property of an object in linked database is semantically overloaded:

When did Germany join the EU?

(connected via dbp:accessioneudate property)

**Granularity of data** is an opposite case to the granularity of language. Sometimes language expression power involves multiple semantic links between objects and cannot be mapped straightforward to a property. Example:

Who was the grandfather of Ashoka?

(double connection via P40 property)

**Filtering of resulting objects** is expressed in natural language by quantifiers, comparative expressions, cardinals or superlative [57]. This edge cases should be handled by deriving filter options from the input phrase structure and by handling special non-textual data types in knowledge bases. Example:

What is the second largest city in the USA?

**Ambiguity** occurs on both sides of mapping process. Some questions in natural language are not straight for human to answer. Object in linked data structure might be linked through multiple links with redundant properties. Common sense allows human to understand which objects with similar labels are semantically acceptable. Question answering systems usually use ranking of answers or mapped objects to estimate how correct produced results are. Also we should note that semantically right expressions in natural language entail cases with an extreme ambiguity.

## Question answering systems architecture

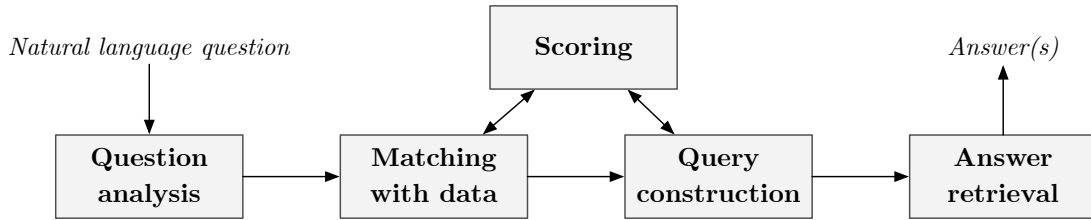


Figure 2.1: Common components of a question answering system.

**Question analysis** step detects a type of the question and parameters of the searched entity. Using linguistic analysis methods a question answering system recognizes Named Entities and classifies the question category.

**Matching with the data** step links input entities with appropriate entities in the knowledge graph. Natural language techniques are used to search by synonymous labels. Indexing tools are used for a textual search over linked data entities.

**Query constructing** step is different for various question answering systems. The step traditionally produces a SPARQL query.

**Answer generation** step applies formed query to the knowledge base and receives a set of entities. After evaluating result entities, question answering system displays the result in requested form (links to the entities/natural language/media enriched data).

## Question answering approaches

According to [57] in this section we provide classification of approaches applicable to the question answering over linked data.

**Approaches based on controlled natural language** consider a subset of natural language phrases that can be processed by the question answering system. Evident pros of this approach is clear representation of input query. Typical example of this approach is GINSENG (Guided Input Natural Language Search Engine). GINSENG relies on a simple question grammar which gets dynamically extended by the structure of an ontology to guide users in formulating queries in a language seemingly akin to English [15].

**Approaches based on formal grammars** typically based on linguistic grammars that assign a syntactic and semantic representation to lexical units and exploit the principle of compositional semantics to compute and overall semantic representation of a question by combining the meaning of the parser as specified in the grammar [57]. ORAKEL is an example of system, which uses this approach. It computes wh-based questions as logical query form and knowledge is represented with F-Logic and Onto broker form. This system is used to convert question into query form, and the given query is fed to bottom-up generalization model for getting intentional answer to the user. Inference engine is used to evaluate queries to knowledge base form. Customization is performed through the user interaction. Linguistic argument structures, such as verbs or nouns with their arguments are mapped to the relations in the ontology [36].

**Mapping linguistic structures to ontology semantic structures** is another approach used by PowerAqua question answering system. In a first step, the linguistic component analyses the NL query and translates it into linguistic triple form. In a second

step the Ontology Discovery sub module of PowerMap, identifies the set of ontologies likely to provide the information requested by the user. To do so, it searches for approximate syntactic matches within the ontology indexes, using not just the linguistic triple terms, but also lexically related words obtained from WordNet and from the ontologies, used as background knowledge sources. After this process, PowerMap 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. In a third step the Triple Similarity Service module takes as input the previously retrieved Entity Mapping Tables and the initial Linguistic triples and extract, by analyzing the ontology relationships, a small set of ontologies that jointly cover the user query. The output of this module is a set of Triple Mapping Tables. Each table relates a linguistic triple with all the equivalent ontological triples. Using these triples the information drawn from the relevant semantic sources is analyzed to generate the final answer [60].

**Template-based approaches** are represented by LODQA and TBSL systems. That kind of systems is constructing a template of the query by a result of question analysis. At the next stage this template is filled up with the results of entities matching. This approach is dataset-sensitive to natural language structural mismatches, which can be caused by natural language granularity. [57]

**Graph exploration approaches** process a question query by selecting a basis entities and processing through the knowledge base graph using matching with the natural language terms. Like any other graph exploration algorithm, this approach is limited to the depth of the graph extension over large amount of linked data. Implemented heuristic allows to efficiently explore graph even without the knowledge of the schema. Typical examples of this approach are Treo question answering system, Top-k exploration approach and the approach by Ngonga et al [57]. Quantified question is another challenge for the graph exploration approach as long as semantic connection doesn't contain trivial evidence of filtering or quantifying.

## 2.6 Relevant applications and technologies

### QALD systems

**FRyA** is NLI to ontologies which balances between heavy customization (which is usually required by application developers, in order to port the NLI system to a different domain), and the end-users who need to explore the available knowledge without being constrained with the query language. FRyA combines the output of the syntactic parser with the ontology-based lookup in order to approve the user's information need and, if necessary, engage the user into the dialog [24].

**PowerAqua** is a multi-ontology-based Question Answering (QA) system, which takes as input queries expressed in natural language and is able to return answers drawn from relevant distributed resources on the Semantic Web. In contrast with any other existing natural language front end, PowerAqua is not restricted to a single ontology and therefore provides the first comprehensive attempt at supporting open domain QA on the Semantic Web [39].

**CASIA** is a question answering system over Linked Data (DBpedia), which focuses on construct a bridge between the users and the Linked Data. Based on the Linked Data consisting of subject-property-object (SPO) triples, each natural language question firstly is transformed into a triple-based representation (Query Triple). Then the corresponding

resources in DBpedia, including class, entity, property, are mapped for the phrases in the query triples. Finally, the optimal SPARQL query is generated as the output result. CASIA can not only deal with the single-relation questions but the complex questions containing multi-relations. CASIA was evaluated on QALD-3 test data set and achieved an F-measure score of 0.36, an average precision of 0.35 and an average recall of 0.36 over 99 questions [32].

**Xser** is a question answering system over Linked Data (DBpedia), converting users' natural language questions into structured queries. There are two challenges involved: recognizing users' query intention and mapping the involved semantic items against a given knowledge base (KB), which will be in turn assembled into a structured query. Authors propose an efficient pipeline framework to model a user's query intention as a phrase level dependency DAG which is then instantiated according to a given KB to construct the final structured query. They evaluate the approach on the QALD-4 test dataset and achieve an F-measure score of 0.72, an average precision of 0.72 and an average recall of 0.71 over 50 questions [63].

**YodaQA** is a question answering system over Linked Data. The QA task is implemented in YodaQA as a pipeline that transforms the question to a set of answers by applying a variety of analysis engines and annotators [13]. It is composed from largely independent modules, allowing easy extension with better algorithms or novel approaches, while as a fundamental principle all modules share a common end-to-end pipeline. The YodaQA pipeline is implemented mainly in Java, using the Apache UIMA framework. YodaQA represents each artifact as a separate UIMA CAS, allowing easy parallelization and straightforward leverage of pre-existing NLP UIMA components; as a corollary, authors compartmentalize different tasks to interchangeable UIMA annotators. Extensive support tooling is included within the package [12].

**Watson/DeepQA** is a software architecture for deep content analysis and evidence-based reasoning. The DeepQA architecture views the problem of Automatic Question Answering as a massively parallel hypothesis generation and evaluation task. As a result DeepQA is not just an answering system – rather it can be viewed as a system that performs differential diagnosis: it generates a wide range of possibilities and for each develops a level of confidence by gathering, analyzing and assessing evidence-based on available data. With a question, a topic, a case or a set of related questions, DeepQA finds the important concepts and relations in the input language, builds a representation of the user's information need and then through search generates many possible responses [54]. For each possible response it provides independent and competing threads that gather, evaluate and combine different types of evidence from structured and unstructured sources. It can deliver a ranked list of responses each associated with an Evidence Profile describing the supporting evidence and how it was weighted by DeepQA's internal algorithms [28].

## 2.7 Trends and open topics

**Combining structured and unstructured data**, as it was performed by IBM Watson, is an efficient approach to answer other than Factoid question. A lot of information is still available only in textual form, both on the web and in the form of labels and abstracts in linked data sources. Therefore, approaches that can not only deal with the specific character of structured data but also with finding information in several sources are needed. These approaches should be able to process both structured and unstructured information, and combine such gathered information into one answer [58]. The integration of structured

and unstructured data gets benefits of both structured and unstructured resources and approaches. Combining available resources mutually enriches the answering process. Current evaluation campaigns include a task on hybrid question answering. Systems retrieving answers for questions that required the integration of data both from RDF and from textual sources.

**User interaction** allows question answering systems to collect a feedback, which is especially important for the machine learning purposes. Allowing for a question answering dialog instead of single questions and answers would allow users to pose questions in a dialog context, e.g. referring to previous questions or answer. Furthermore, the utilization of interaction modalities may improve the efficiency of the question answering process, in particular in a dialog context [57].

Besides, the following techniques gain a traction:

- **Confidence measurement** can provide more feedback for the end-user and allow to tune search techniques for the particular question.
- **Interlinking web of data** is a key trend in development of question answering systems. General domain knowledge bases can supplement each other to answer a complex question.
- **Machine learning** becomes more applicable to natural language processing after new researches in RNN (recurrent neural network) and word embedding. Question answering over linked data just starting to be considered as a machine learning task.



# Chapter 3

## Design of the system

The chapter describes principles of the proposed question answering system. Each approach described in section 2.5 has a segment of questions with a high quality of achieved results. To form a list of requirements for our system, we are going to select cases and challenges. Afterwards, the section describes the proposed approach of solving these challenges and reveals the approach using high-level scheme and the algorithm. Consequent result of proposed approach is a list of key features and known constraints. To explain the proposed approach principles, examples of sentences processing are provided in the end of the chapter.

### 3.1 Requirements

General domain solution can be applied for business interests and regular question answering search. Most of semi-structured companies' data can be transformed into regular knowledge representation formats (such as RDF), which are used by general domain prominent knowledge bases. General domain solution, which has a low entry barrier to extend the system, can be used in the commercial sectors of the natural language answering industry. Due to the data availability, our system should be developed over general domain knowledge bases. Also, the system should be easily scalable to private datasets.

Make applicable solution is a high-priority aim for the project. A commercial usage integrated with chatbots and search purpose require the system to provide a natural end-user experience, instead of providing links to the knowledge base entities. Final solution should be flexible to integrate into existing conversational agents, dialog systems or any natural language interfaces. Scenarios mechanism integrated to the question answering system should combine dialog system components to provide better user experience.

Schema-agnostic approach was selected as a basis of our study. Prominent linked data knowledge bases, such as DBpedia and Wikidata, are not unified. Extraction of valuable knowledges require user (software) to use service-specific queries. Schema-agnostic approach, which uses only primary features of linked data, makes the system more universal. Moreover, existing companies' data potentially can have a complex structure in comparison to general domain knowledge bases. Feasible solution can be built upon the basis of triples with the usage of less features.

Question granularity and answers granularity obstruct classical methods of mapping natural language lexemes to SPARQL query structures. In our work we are trying to scale graph comparison methods to linked data. It becomes possible by searching of a semantic connection between noun phrases and named entities in input and output phrases. Whole

system is based on the following assumption: “if a known question has a correct and complete known answer, we can recognize an input question as a similar and produce an answer with the same semantic links over the knowledge base”. Searching of semantic links in the knowledge base is a core of developed system. Semantic links are represented as a set of paths over directed graph of dataset. They can be stored and applied to any item of the dataset. Complete description of semantic links search is provided in section 3.5.

Matching input sentences with known questions is one of the tasks for our system. Syntactic comparison of sentences is based on our previous work related to dialog systems [18]. The question answering system was developed from scratch. System should be capable to match sentences with distorted syntax tree structure and sentences with included redundant natural language items. In addition, high-priority objective is tuning of the semantic module for every type of questions. Every type of questions discussed in section 2.1 requires corresponding methods to optimize question answering process.

Focus on the Wikidata knowledge base is one of our objectives. Last couple years Wikidata project receives attention despite of the remaining DBpedia popularity<sup>1</sup>. Answering over Wikidata was included into the QALD tasks in year 2017. Wikidata has granular properties in comparison to DBpedia. Granular properties are much compliant to the proposed graph-based approach.

Entities extraction should not be limited to named entities extraction, but should include every noun phrase available in the answer. Transforming available noun phrases into possible permutations and substituting words with possible synonyms provided by thesaurus improve a rate of linkage with the data. The step of matching language entities with data is general to every question answering system. Our system should combine best practices to maximize the rate of matching.

## 3.2 Processing pipeline

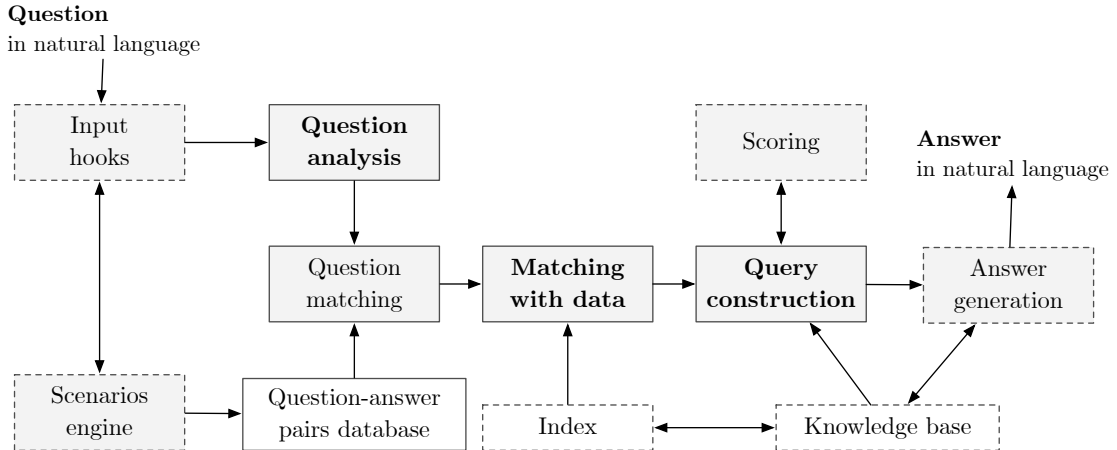


Figure 3.1: Designed processing pipeline of the question answering system.

<sup>1</sup>Source: <https://scholar.google.com>. Wikidata: 670 citations in 2016 (+97%). DBpedia: 3140 (+12%).

Figure 3.1 shows key elements of the designed system processing pipeline. *Input hooks* block binds input channel with one of the available interfaces (console, WebSocket, IM, etc.). In case of scenario control, *scenario engine* takes the control and asks questions to a user, working as a dialog system. If scenario engine figures out an answer to the question, module is going to update *question-answer pairs database*. That database accumulates phrase syntactic trees and semantic relations between question and answer entities. In case of regular request *question analysis* module recognizes a type of the question. Later, *question matching* block compares syntactic tree of the input phrase with known candidates and uses a knowledge base index to match noun entities properly. Semantic query is performed over the knowledge base. *Answer generation* engine produces the answer in natural language by matching the answer template with received entities.

### 3.3 Analysis of features

#### Key features

Schema-agnostic graph-based approach entails resistance of the question answering system to granularity issues. Granularity of data appears in graph-based approaches as a requirement of long connections between nodes. The question answering systems is able to create links within question and answer entities. This technique creates preconditions inside of input questions and postconditions inside of output questions. It is another way to improve the rate of correct matching.

The designed question answering system includes dialog system components to perform scenarios while receiving user's feedback. This design also provides a possibility to recognize an unknown answer in opposition to an unknown linked data connection. Dialog unit of the designed question answering system should be able to ask user for an unknown answer and derivate a semantic connection from question-answer pair, while extending a set of acceptable questions. Natural interface of the system extension makes possible improvement of the system by several user roles. We can easily involve end-users into the process of the system improvement with the aid of scenarios.

Current dialog system is designed for English language, but graph-based approach do not restrict us to apply algorithms to any localized items. Natural language processing unit abilities will allow the whole system to be able to process new languages.

#### Known constraints

In comparison to traditional approaches, which focus on constructing SPARQL queries our technique has an obvious constraint. Question answering requires answers of questions to improve itself. Answering question without provided reference question was implemented. That process isn't an advantage of the system and was created only for evaluation using standard metrics. Semantic relations have a form of paths over linked data graph. They allow us to predict a straight number of items, which are eligible to match the reference question. That evaluation is provided in the section 5.2.

Filtering questions processing have an experimental nature. The only option available to graph-based approach is to inference attributes of filtering from the semantic connection. Inference might be acceptable with the small amount of results. It becomes impossible to inference a correlation between all properties in general listing questions. That constrain shows us insufficiency of exclusive items linking and importance of properties linking.

The special techniques for definition and explanation questions are required. Definitive questions are handled by matching with the link data items. The next step is a looking for a source of information and requesting textual data from the source (primary source is Wikipedia). Explanation questions stay at the field of experiments.

The question answering without reference has certain issues with links of the one property. These semantic links don't contain any other noun phrases to derive claimed property. It can be solved by processing and linking properties with the language entities in a similar manner as the previous issue.

### 3.4 Algorithm description

The processing flow of the designed question answering system combines following multiple algorithms:

- matching with data;
- answering with a reference;
- extending known question-answer pairs;
- answering without a reference.

Algorithm 1 parses syntax tree of the sentence, extracts noun phrases and links with entities of the knowledge base. Algorithm 2 tries to answer using known question. As for general domain, it is unlikely that the reference is going to be available. Question answering systems with a specific domain are able to cover a large set of questions and provide conceived answers using this method.

```
Input: question, characters sequence  
Output: syntax tree, matched dataset entities  
tokenize sentences and classify the language of the question;  
parse syntax tree structures;  
foreach noun phrase in the syntax tree do  
|   create noun phrase permutations;  
|   try to match permutations with knowledge base entities;  
end
```

**Algorithm 1:** Matching with items of a knowledge base.

Algorithm 4 (available at the end of this section) will be applied in case of unavailable reference. Regular use case of the designed approach is known pairs extension. Algorithm 3 represents providing question-answer pair. The result of this process is an addition of the new reference including a semantic path to the database. The proposed system uses a graph-based schema-agnostic approach, but also can be classified as a template-based due to the initial syntactic tree comparison.

### 3.5 Processing examples

The section provides in-depth overview of input processing of the designed question answering system. Each use case was illustrated by an example of a question and appropriate steps description.

```

Input: syntax tree, matched dataset entities
Output: natural language answer
foreach reference question in pairs database do
  if similarity > threshold then
    apply semantic path to items using SPARQL;
    if applicable then
      construct answer by substituting item labels;
      return answer
    else
      /* answer isn't available in the dataset
    end
  end
end

```

**Algorithm 2:** Answering with the use of a reference question.

### Matching with data

String (characters sequence) is an input form of the system. At the first stage the system checks the number of sentences in provided sequence using NLTK. The system splits an input query to separate sentences. Our system rejects queries with multiple sentences inside. Processing of multiple sentences, with a context saving is related to conversational agents and dialog management. That kind of features were ignored at the moment. Further, the system is going to check the language of the sentence using external module (see section 4.4 for details). Used linked datasets are multilingual, but our system accepts only English questions. If an input string wasn't rejected, it moves into the syntax tree parser. The system uses spaCy as a primary NLP unit.

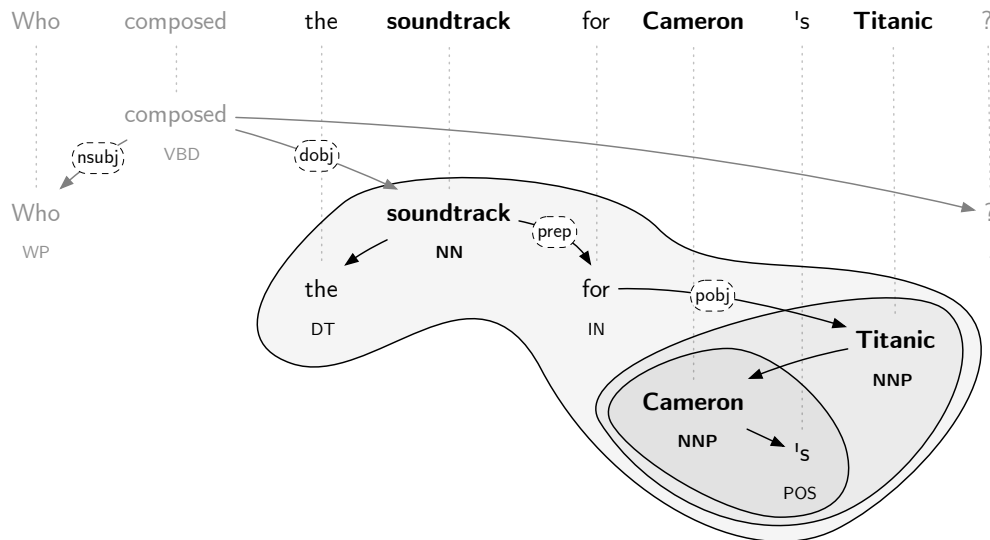


Figure 3.2: Selecting noun subtrees from the syntax tree.

Figure 3.2 shows an example of the parsed input sentence. Relations between words of the sentence are represented by arrows with a corresponding relation type on labels. After

**Input:** question and answers syntax trees, matched dataset entities  
**Output:** semantic path

```

foreach question's entity do
  | foreach answer's entity do
  | | foreach possible connectivity direction do
  | | | find a connection using SPARQL;
  | | end
  | end
end
score and sort found paths;
while user do not accepting meaning do
  | ask about pre- and post- conditions;
  | generate an example of a linkage;
  | show semantic path and example;
end
save semantic path to the pairs database;

```

**Algorithm 3:** Extension of the question-answer pairs database.

receiving syntax tree, the system starts the process of entities extraction. This process is a necessary step for every use case described below. In case of successful reference matching the system is required to check the ability to find suitable entities to apply the semantic path.

Current system is focused on noun entities. In comparison to other parts of speech nouns have a straightforward related element on the linked data graph — nodes. As opposed to nouns, edges mean verb or adjective entity. The system searches for an appropriate tag mark to extract noun phrases from a sentence. SpaCy uses part-of-speech tags used in the Penn Treebank Project and tags are available at the figure 3.2 under each word. Following tags are related to the root of a noun phrase: NN, NNS, NNP, NNPS. The sentence from the previous example contains three nouns: soundtrack, Cameron, Titanic.

For each noun occurrence our system extends a word to a subtree. Rounded shapes at the figure 3.2 represents found subtrees. These subtree may include other subtrees, but all intersections will be skipped at the later steps of the processing pipeline.

To increase quality of linking with the knowledge base entities, the system creates permutations of founded phrases. Initial set of three noun phrases with removed inclusions, extends by all of the subtree variations as shown at the left side of the figure 3.3. We should note, these structural permutations always save the root word. It is required to produce syntactically right language items, which can be recognized as noun phrases.

Extra permutations are created using WordNet thesaurus. For every word in noun phrases set, the system will generate variations of substitution with a synonym. WordNet allows the system to search only synonyms with the same part-of-speech inside a synset to save the initial meaning. The module generates synonymic permutations for every generated structural permutation as shown on the right side of the figure 3.3.

The system requests items URIs from knowledge bases using created set of noun phrases. Wikidata uses IDs formatted as follows: Q42, where 42 is the number of the item. DBpedia uses labels to reference an item of the dataset. At the listing 3.1 you can see items' IDs

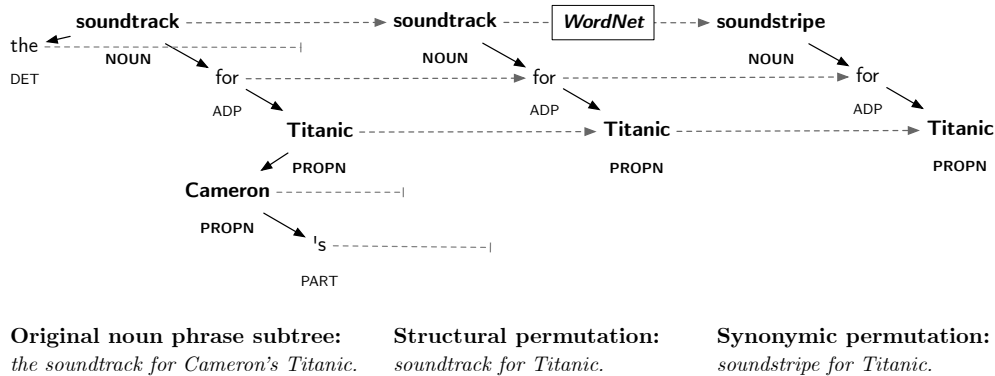


Figure 3.3: Structural and synonymic permutations of the subtree.

bounded to noun phrases extracted from the input query (count of relevant items is limited to 3 at the example).

Listing 3.1: Result of the matching with data stage.

---

```

1 <ENTITY_SET>
2 <ENTITY> the soundtrack for Cameron 's Titanic (3)
3   <BATCH> soundtrack for (10)
4     (Q28449220, -) (Q7564988, -) (Q16950640, -)
5   <BATCH> the soundtrack (7)
6     (Q15749736, -) (Q2165175, -) (Q19867933, -)
7   <BATCH> soundtrack (10)
8     (Q217199, -) (Q7564979, -) (Q7564980, -)
9
10
11 <ENTITY_SET>
12 <ENTITY> Cameron 's (1)
13   <BATCH> Cameron (10)
14     (Q3971976, -) (Q225123, -) (Q959820, -)
15
16
17 <ENTITY_SET>
18 <ENTITY> Cameron 's Titanic (1)
19   <BATCH> Titanic (10)
20     (Q25173, -) (Q44578, -) (Q738840, -)

```

---

Search results may contain redundant items despite of the ordering by relevance. As it turned out reducing the number of items is important at this stage. Large amount of items lead to exponential amount of possible connections during the unknown questions answering. Depending on a strictness settings the system rejects up to every non-identically labeled item and items without the highest relevance. Entities set with intersections in linked knowledge bases items will be merged after that reduction step. That step is important because certain processing elements will find shortest connection between entities

sets. If sets weren't properly merged, it might cause production of short semantic paths. Merged set of entities is shown at the listing 3.1.

Listing 3.2: Example of merged entities.

---

```
1 <ENTITY_SET>
2 <ENTITY> Pierce (1)
3   <BATCH> Pierce (10)
4     (Q225330, -) (Q1517751, -) (Q963193, -) (Q6453166, -) (Q10863000, -)
5
6 <ENTITY> Pierce Brosnan (2)
7   <BATCH> Pierce Brosnan (1)
8     (Q81520, -)
9   <BATCH> Brosnan (2)
10    (Q4975406, -) (Q19089805, -)
```

---

Created entities sets are used as a linkage between a sentence and the knowledge bases items at the later processing steps.

### Answering using reference question

Answering question with a reference question available in the database is a basic and primary use case of the system. One of input of the system is a database which contains triples of question tree, semantic path, answer tree. Detailed structure of the data is provided at the listing 3.3.

Listing 3.3: Item of the known question-answer pairs database.

---

```
1 {
2   "question": b" ... ", // binary serialized question's syntax tree
3   "answer": b" ... ", // binary serialized answer's syntax tree
4   // bindings to a noun entities
5   "items": [
6     {
7       "origin": "question",
8       "token_id": 72836,
9       "match": {
10        "dataset": "wikidata",
11        "id": "Q889821"
12      }
13    },
14    ...
15  ],
16  "solution": [
17    {
18      "dataset": "wikidata",
19      "semantic": ["P279", "Q880198", "P39", "Q1293842", "P27"],
20      "strict": false
21    },
22    ...
23  ],
```



```

24  "precondition": [
25    ...
26  ],
27  "postcondition": [
28    ...
29  ]
30 }

```

Question matching process starts by comparison of input question with every available question from the database. Every reference question receives a score, derived as a percentage of similar links inside of tree. This comparison involves lemmas of the words if none of the word isn't part of a parsed noun phrase. This condition guarantees similarity of trees and allows variability at the position of a noun phrase at the same time. An example of trees matching with evaluation of possible variability available in the section 5.2.

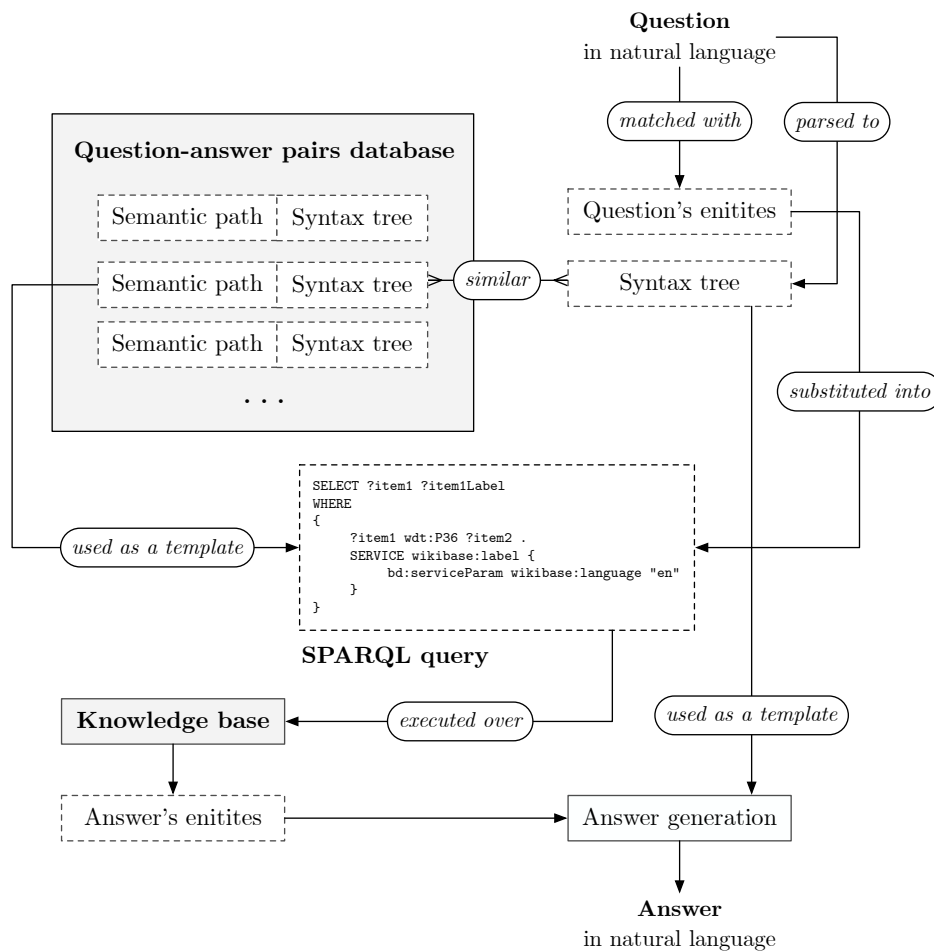


Figure 3.4: Answering with a reference scheme.

Question sentence with a highest similarity score, which surpasses the manually set threshold is used as a reference sentence. Every noun phrase of an input question links to a proper noun phrase of a reference question. Semantic path in the database of known question-answer pairs contains a list of properties, which are applicable to the items. The

system tries to apply that pass by formulating a SPARQL queries for every linked item from an input question (shown at the listing 3.4). Configuration of the question answering system has an option, which determines a behavior in case on unmatched entities in the output phrase (ignoring, rejecting).

Listing 3.4: Example of a SPARQL query generated using semantic path.

```

1 SELECT ?item3
2 WHERE
3 {
4   wd:Q6425 wdt:P176 ?item2.
5   ?item2 wdt:P159 ?item3.
6   SERVICE wikibase:label { bd:serviceParam wikibase:language "en" }
7 } LIMIT 100

```

Question recognized as answered in case of application at least of one path, which connects item from (and a corresponding noun entity) to another item from reference answer. Depending on desired result, question answering system can perform question constructing by inserting output item's labels into reference answer. The system also tries to adjust word configuration to the same as noun phrase from reference answer has. Eventually, the output answer moves to the output of the systems and can be processed by connected interfaces while passes to the user.

### Extending question-answer pairs database

Knowledge base with items described at the listing 3.3 is a core of the system. Extending of the database with semantic links extends a number of acceptable by the system questions. To extend the database, user with assigned administrative role provides a question-answer pair to the system. System tries to find a semantic connection between question and answer by searching over the linked data graph. Question answering system tries to connect every element of the question's entities sets to another element from the answer's entities sets. Scheme of the process is provided on the figure 3.5.

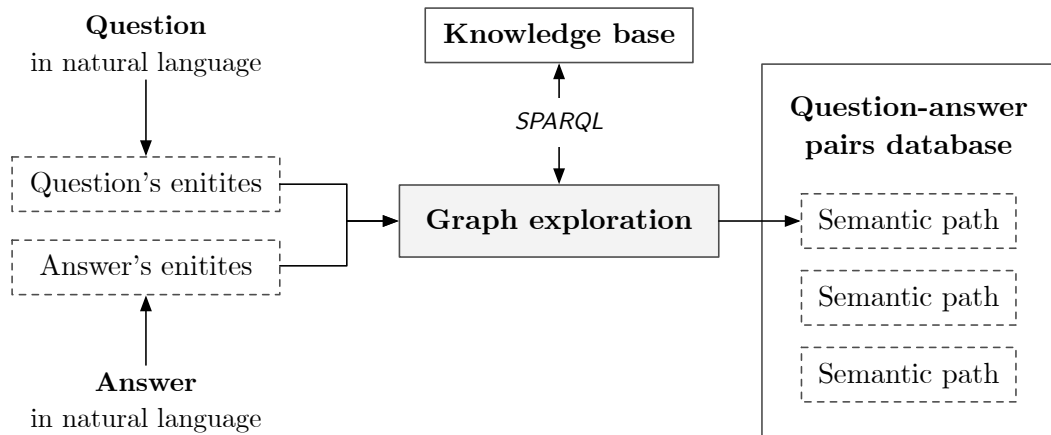


Figure 3.5: Extension of the question-answer pairs database scheme.

Graph traversal stops, when the system found a path (s), which connects every answer's entity with one entity from the question. At the low level of abstraction linking of two items in one knowledge base is performed querying SPARQL endpoints. RDF triples create a directed linked data graph, yet semantic relation isn't directional thing. SPARQL querying is an optimal and universal method of search through the linked data graph. The question answering system generates possible connection direction and gradually increases acceptable length of a path. Listing 3.5 shows SPARQL query examples to receive the following semantic path:

(x) -[ wdt:P176 ]-> ( \_ ) -[ wdt:P159 ]-> (x)

Listing 3.5: Example of a SPARQL query that searches for a semantic path.

---

```

1 SELECT ?prop1 ?item2 ?prop2
2 WHERE
3 {
4     wd:Q6425 ?prop1 ?item2.
5     ?item2 ?prop2 wd:Q1297.
6     SERVICE wikibase:label { bd:serviceParam wikibase:language "en" }
7     FILTER ( !strstarts(str(?prop1), "http://wikiba.se/ontology") )
8     FILTER ( !strstarts(str(?prop2), "http://wikiba.se/ontology") )
9 } LIMIT 100

```

---

The system allows a user to check the correctness of a found path not only by checking a syntax tree structure, but also by checking relations between the entities inside of a sentence. That pre-conditions and postcondition are semantic paths, which are executed before an acceptance of an input question or an output answer. Following example of a question requires a precondition:

In what city is the Heineken brewery?

While system applies a semantic path, with manufacturer (P176) and headquarters location (P159) properties, the Heineken (Q854383) item can be substitute only by brew label:

In what city is the Coca-Cola brewery? (invalid)

Relation between Heineken (Q854383) and brewery (Q131734) items via Heineken International (Q180855) is required.

### Answering an unknown question

Regular question answering systems generates SPARQL queries based on a natural language sentence and request the query to a knowledge base. These systems do not use any reference while preparing known answers. Proposed approach requires a set of known question-answer pair to derive semantic links. To make or solution comparable with other question answering systems using standard metrics, we implemented a flow for a question without known answer.

When the system decide, that question reference isn't available in the database, the process of answering without a reference begins. First step of the process is to classify

a question type. The system searches for a corresponding syntax tree substructures of a certain question types.

Noun entities linked with the knowledge bases' items is an input of the answer deriving process. That process uses an assumption that output entity (at least in case of a single factoid question type) is a node of one of the paths between entities. The system searches for paths over knowledge base between each possible entities pair from an input sentences. An example of found paths is provided on the figure 3.6 .

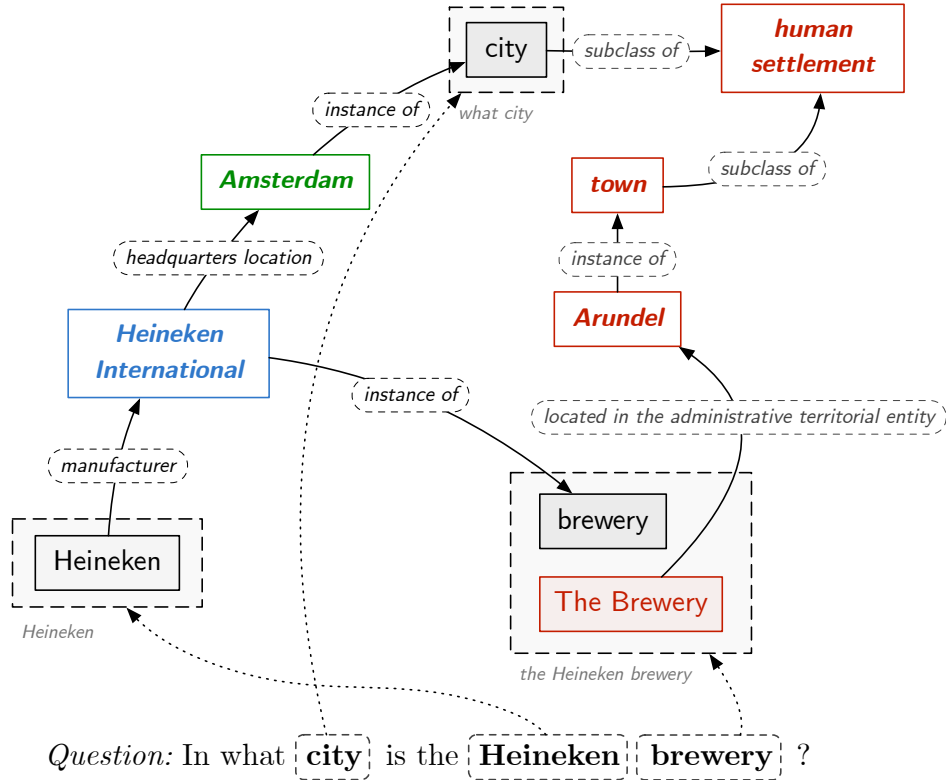


Figure 3.6: Answering an unknown question over linked data graph.

Figure 3.6 contains a fragment of the dataset with semantic paths between every question's items. Connection of knowledge base items built the same way as discussed earlier in the previous subsection. After creation of paths list, the system filters them.

One of filters examples is a symmetrical filtering. Large subset of knowledge base's items can be connected through a general. Occurrence of a general item which connects a large set of items can be detected by a symmetric property occurrence in a semantic path. The following example shows how Sparta (Q5690) and biophysics (Q7100) are connected via Armenian Soviet Encyclopedia (Q2657718):

```
( wd:Q5690 ) -[ wdt:P1343 ]-> ( wd:Q2657718 ) <-[ wdt:P1343 ]- ( wd:Q7100 )
```

Scoring process depends on a question type. In the case of a factoid question with a single answer, the system searches for an item which occurs in maximal number of connected paths. Individual paths have a higher score if they have unique properties inside of group. Factoid questions with multiple answers as opposed to questions with a single answer have another scoring priorities. The system searches for items with surrounding elements that consistently persist in a set of found paths. As for boolean questions, we implemented a flow, that recognizes statement as a true if exist unique path between elements with a shorter length, than other found paths. Time and value questions are handled the same way as in case of factoid question with a higher scoring of date/time and numerous values.

Result of that answering process is a sorted list of knowledge base instances, which doesn't allow the system to construct a natural language answer. System output in case of embedded solution is basically a label of the item with a highest score.

**Input:** syntax tree, matched dataset entities  
**Output:** datasets items

classify the question type;  
**foreach** *question's entity* **do**  
  | **foreach** *other entity from the question* **do**  
  | | **foreach** *possible connectivity direction* **do**  
  | | | find a connection using SPARQL;  
  | | **end**  
  | **end**  
**end**  
**if** *question type is boolean* **then**  
  | **if** *every item is linked* **then**  
  | | **return** *true*  
  | **else**  
  | | **return** *false*  
  | **end**  
**end**  
**if** *question type is factoid* **then**  
  | **if** *multiple answers required* **then**  
  | | score items lying on paths;  
  | | **return** [*set of items*]  
  | **else**  
  | | remove symmetrical paths;  
  | | score items lying on paths;  
  | | **if** *answer type is resource* **then**  
  | | | **return** *item with the highest score*  
  | | **else**  
  | | | score candidates' properties;  
  | | | **return** *item with the highest score*  
  | | **end**  
  | **end**  
**end**  
**return** *answer is not found*

**Algorithm 4:** Answering without a reference question.

# Chapter 4

## Implementation

Python was selected as a programming language for the implementation. High-level programming language with a dynamic typing is an appropriate tool for natural language processing. Question answering systems are not environment-dependent solutions. Developed system is cross-platform: it was tested at the Linux operation system. The distribution includes prepared python environment, which automates set up process. Complete setup instructions are available in Appendix B. Generated technical documentation and UML class diagram were prepared for future development and maintenance. Source code is licensed under the MIT license and distributed at the GitHub repository<sup>1</sup>. In addition the code base follows PEP8 guidelines and passes static analysis.

### 4.1 Architecture

Architecture of the question answering system is shown on the figure 4.1. That modular view scheme in comparison with the figure 3.2 describes how individual components of the system work and communicate to compose the final pipeline. Questions for the subsequent answering or question-answer pairs generation are inputs of the system.

These questions move through the entities extraction module to create tree structures linked with data. Those questions are switched between core functionality and scenarios engine in case of an interactive extension. The core functionality module extensively uses graph connection module with appropriate bindings to the Wikidata and the DBpedia endpoints. As a result the system produces answer(s) in form of a natural language or in the form of items set.

### 4.2 Modules

Question answering system contains following modules:

- `__init__.py` package wrapper
- `__main__.py` console utility functions
- `common.py` common functions
- `core.py` answering and extension
- `dbpedia.py` DBpedia SPARQL and JSON endpoints bindings
- `graph.py` semantic search over a knowledge graph

---

<sup>1</sup>Link: <http://github.com/kusha/qas>.

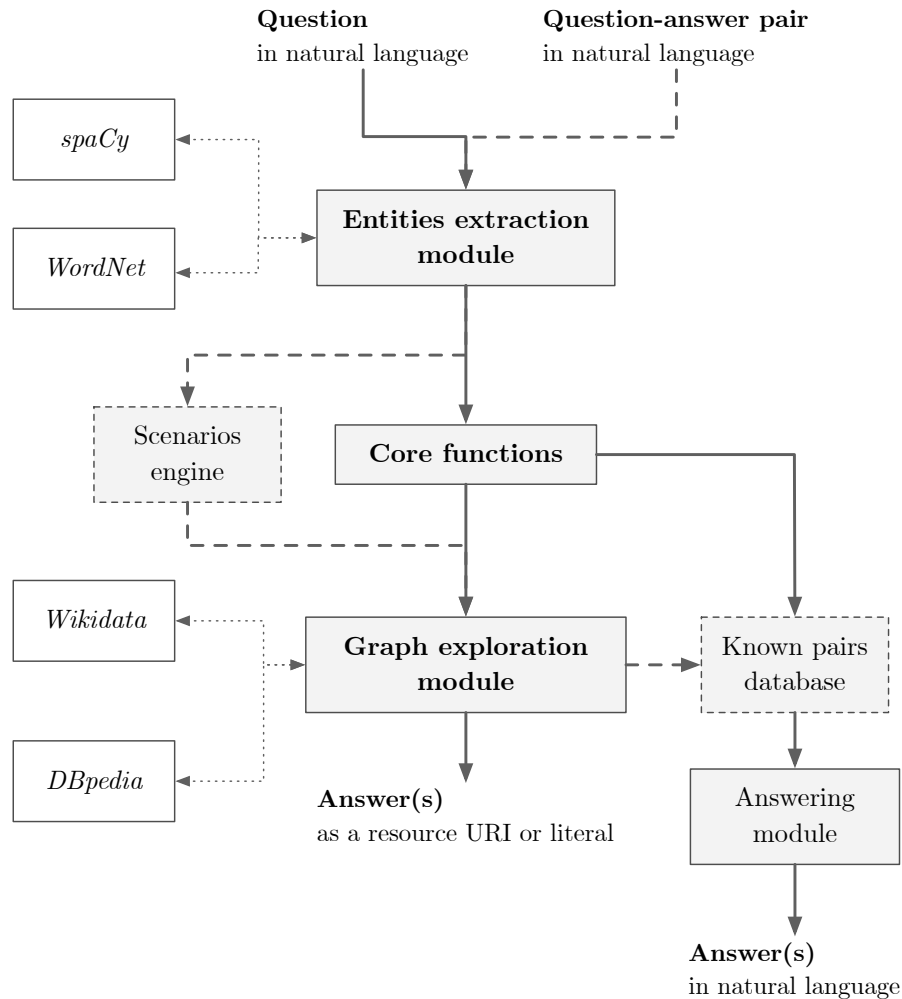


Figure 4.1: Architecture of the developed question answering system.

- *items.py* entities representations
- *link\_grammar.py* Link Grammar Parser bindings (deprecated)
- *logs.py* shared logging module
- *noun\_phrase.py* matching with data
- *scenarios.py* scenarios interpreter
- *sentence.py* natural language processing unit
- *wikidata.py* Wikidata SPARQL and JSON endpoints bindings

### 4.3 Linked data integration

DBpedia and Wikidata are main datasets for the project. The system uses a combination of a SPARQL querying and JSON requests for both knowledge bases. JSON endpoint of knowledge bases is flexible, but the way of the information retrieval is less efficient. Several heuristics were tested to figure out a proper way of a BFS (breadth-first search) over a dataset. Searching of a connection between two entities is more efficient by querying



a SPARQL endpoint with a connection of different lengths. The reasons are optimized search engines (see more implementation details in section 2.3).

Wikidata is requested using *MediaWiki API* endpoints and *Wikidata Query Service* SPARQL endpoint.

### MediaWiki API endpoints

MediaWiki is a free software open source wiki package written in PHP, originally for use on Wikipedia. Now it is also used by several other projects of the non-profit Wikimedia Foundation including the Wikidata project. The MediaWiki action API is a web service that provides convenient access to wiki features, data, and metadata over HTTP. Clients specify an action parameter to request particular “actions”.

**wbsearchentities** action searches for entities using labels and aliases. The action returns a label, description for the entity and details of the matched term in the user language if it is possible. The matched term text is also presented in the aliases key if different from the display label.

**wbgetentities** action gets the data, including labels and relevant links for multiple Wikibase entities.

### Wikidata Query Service

The Wikidata Query Service provides SPARQL endpoint and GUI interface<sup>2</sup>. This endpoint allows to request data in the SPARQL RDF query language. With SPARQL we are able to extract any kind of data represented in JSON, with a query composed of logical combinations of triples.

### DBpedia Lookup

DBpedia Lookup used to linking entities with the data. DBpedia is able to return JSON data instead of render HyperText Markup Language (HTML) pages. Public DBpedia SPARQL endpoint is used to search for semantic paths.

The DBpedia Lookup Service can be used to look up DBpedia URIs by related keywords. Related means that either If the label of a resource matches, or an anchor text that was frequently used in Wikipedia matches the service consider related keywords. The results are ranked by the number of inlinks pointing from other Wikipedia pages at a result page.

The **KeywordSearch API** of the DBpedia Lookup was used to find related DBpedia resources for a given noun phrases as strings.

DBpedia provides the access to every entity in the structured JSON format.

## 4.4 Third-party components

### NLTK

NLTK toolkit, discussed in the section 2.4, was used for a basic natural language processing operation such as separate word transformations. Furthermore, NLTK incorporate WordNet thesaurus, so synonymous permutations are generated using NLTK [6].

---

<sup>2</sup>Wikidata Query Service GUI: <https://query.wikidata.org>.

## spaCy

First prototype of the system was built using Link Grammar Parser. Link Grammar Parser was originally used in the work [18] as a parser of syntax trees. Links produced by parsers fits well to the comparison module of the system, but there are two main reasons of the switch to another natural language parses. The first thing is a usage of a non-standard tags and links, which can't be reused in the NLTK module. spaCy uses the Penn Treebank part-of-speech tags as well as NLTK [5]. Dependency-based parse trees produced by spaCy instead of a constituency-based parse trees by Link Grammar Parser. While additional syntactic structure associated with constituency-based parse trees remains an object of debate, reduction in the average in number of nodes is beneficial for tree comparison. Processing speed wasn't an essential reason of this change. Compared to overall processing speed syntax tree parsing time is inconspicuous. Speed of the parsing was estimated for both systems using the prepared evaluation dataset. Link Grammar needs an average of 272.3 ms to parse the sentence, while spaCy needs only 1.6 ms.

SpaCy is written in Cython with a static typization, which allows it to achieve the performance of a native C code. This unit uses Penn Treebank tag as NLTK. Part-of-speech tagging is implemented with the greedy decoding and averaged perception techniques. Shift-reduce dependency parser creates dependency trees 171 times faster than previous implementation with a Link Grammar Parser.

## langdetect

langdetect module uses an adaptation of the vector space model to n-gram to detect language. This module uses a directory of samples language documents to collect all the n-grams up to specified limit. Using created global vocabulary (with associated frequencies) over the documents set it builds a reverse index with a vector of weights calculated with TF-IDF (Term Frequency, Inverse Document Frequency) that represents each file in the space where each dimension is an n-gram. During the language detection it receives a piece of text and return a list of languages sorted in the reverse order by scores of similarity. A text is treated in a similar fashion to the language documents; it is represented as a vector of weights in the n-grams space. Only purpose of the module is to filter non-English queries.

## Requests

Request module is HTTP library for Python that makes possible parallel requests to APIs and endpoints. Support for concurrent requests, powered by gevent library allows us to make parallel request at the matching with the data step and SPARQL querying at the core linking steps [8].

The async module of the requests has the exact same API as requests, except it it returns the Request object instead of sending the request immediately. Further prepared request is executed using `async.map()` method, which also allows us to specify the number of parallel connection and handlers for a specific HTTP errors.

# Chapter 5

## Evaluation

Evaluation is a necessary part of the work which allows us to assess achieved results. The chapter covers statistical measures of the developed system and argues achieved results.

### 5.1 Datasets

#### QALD challenge

In particular, natural language interfaces to online semantic data have the advantage. They can exploit the expressive power of Semantic Web data models and query languages, while at the same time hiding their complexity from the user. However, despite the increasing interest in this area, there was a lack of evaluations that systematically evaluate this kind of systems in contrast to traditional question answering and search interfaces to document spaces. To address this gap, authors have set up a series of evaluation challenges for question answering over linked data. The main goal of the challenge was to get insight into the strengths and capabilities of question answering systems as interfaces to query linked data sources [40]. Another aim is to benchmarking how these interactions can deal with the fact that the amount of RDF data available on the web is very large and heterogeneous. Authors report on the results such evaluation campaigns.

Another goal of the QALD challenge is to evaluate and compare question answering systems with respect to their ability to cope with large amounts of heterogeneous and structured data. Participating systems mediate between semantic data and users who express their information needs in natural language. The main motivation behind QALD is to provide a common evaluation benchmark that allows for an in-depth analysis of the strengths and flaws of current semantic question answering systems. Systematic campaigns allow to track the progress of systems over time. The task for participating systems is to return for a given natural language question and an RDF data source, a list of entities that answer the question, where entities are either individuals identified by URIs or labels, or literals such as strings, numbers, dates, and booleans [40].

For the QALD challenge, authors simplified the evaluation process as much as possible. Participants ran their system locally and submitted the results in an XML file via an online form. Current systems are based on very different infrastructures. For example, they may use the provided SPARQL endpoint or not, they might be based on different semantic database servers (such as Virtuoso), and they may require indexes to optimize performance, as well as different configurations and libraries (such as GATE or WordNet). For the

QALD challenges, authors therefore designed an evaluation with the goal of facilitating participation as much as possible [22].

Participating systems are evaluated with respect to a precision and a recall. Evaluation includes the average time of queries processing. Answers provided by a participating system are compared to the answers provided by the gold standard. The evaluation tool computes precision, recall and F-measure for every question as described at equations 5.1, 5.2 and 5.3 respectively.

$$Recall(q) = \frac{\text{number of correct system answers}}{\text{number of gold standard answers}} \quad (5.1)$$

$$Precision(q) = \frac{\text{number of correct system answers}}{\text{number of system answers}} \quad (5.2)$$

$$F\text{-measure}(q) = \frac{2 \times Precision \times Recall}{Precision + Recall} \quad (5.3)$$

### Prepared dataset

At the moment Wikidata dataset becomes an object of research at the QALD challenge. Focus on Wikidata and a necessity of a reference questionanswer pairs pushing us to creating a new evaluation dataset. It was created by mixing following QALD datasets:

- `qald-6-test-hybrid.json` (*hybrid question answering over RDF and free text data*),
- `qald-6-train-hybrid.json`,
- `qald-6-test-multilingual.json` (*multilingual question answering over RDF data*),
- `qald-6-train-multilingual.json`,
- `qald-7-train-multilingual-extended-json.json`,
- `qald-7-train-en-wikidata.json` (*English question answering over Wikidata*),
- `qald-7-train-hybrid-extended-json.json` (*updated hybrid question answering*),
- `qald-7-train-largescale.json` (*large-scale question answering over RDF*).

Datasets preceded to the QALD-6 were skipped because actual datasets are mixing previous datasets and the gold standard questions set. We extracted all questions from QALD datasets, mixed available DBpedia and Wikidata URIs and completed the missing attributes. That was done by extracting available labels and querying the endpoint for missing attributes.

Radar chart 5.1 characterizes combined dataset. It shows the amount of reused questions. Total amount of questions in the produced dataset is 632. Total number of answers is 1859, with an average of 2.94 answers per question. Factoid questions constitute 70.25% of the dataset, while other answer types (number, date, boolean) all together constitute 29.75%. The newest QALD datasets also provide keywords set for every input question. Keywords containing noun phrases were filtered out (for example question words were filtered). Later, those dataset items were used to evaluate matching with data step of the designed pipeline. It is 526 question with specified keywords in the dataset.

Our system primarily designed for the Wikidata usage, so interlinking of the generated dataset is necessary. 398 factoid question were successfully interlinked. Answers presence in the Wikidata doesn't guarantee the presence of a semantic link or even input phrase entities. Information retrieval is usually done by bots, which discover and extract linked data from a specific domain.

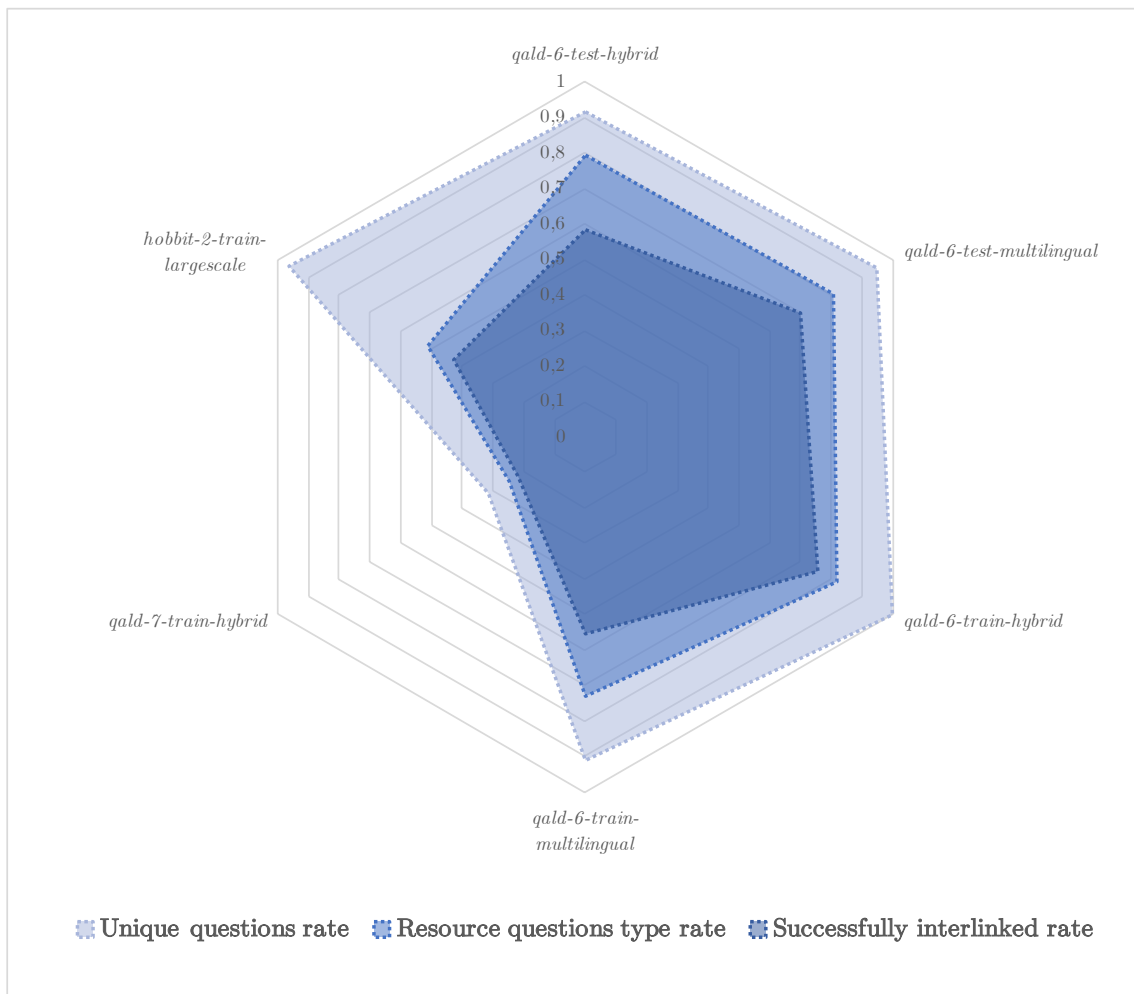


Figure 5.1: Components of the final evaluation dataset.

## 5.2 Evaluation measures

### Answering evaluation

Our system focuses on linking items within a dataset. It uses the following assumption: path over a linked data graph represents semantic relation of a question. Valid path extraction and availability of the data in the dataset are prerequisites of the evaluation.

SPARQL language allows us to calculate possible applicable items for a certain semantic link. That feature was implemented as a part of the system. Possibly it has a high potential in future development. Also, that feature is used to generate a set of possible substitutions while adding known question-answer pair using natural language.

For example:

What is the capital of Austria?

The answer (Vienna, Q1741) will be generate the connection via the property capital (P36). We can estimate a number of possible question-answer pairs by querying the follow SPARQL query:

```
SELECT (COUNT(DISTINCT ?item1) AS ?count)
WHERE
{
    ?item1 wdt:P36 ?item2 .
    SERVICE wikibase:label { bd:serviceParam wikibase:language "en" }
}
```

A total number of 55058 question-answer pairs are available over the dataset. Provided question can be recognized as simple.

Sometimes the system meets a high granularity of natural language and a high-level of abstraction in the dataset. For example, a question from QALD dataset: Who is the president of Eritrea? Item President of Eritrea (Q19108193) is an example of a data granularity and generated path uses that item instead of separate Eritrea (Q986) and president (Q30461) items. In a result, the found semantic path is not useful for future querying.

We should note that evaluation of the answering process takes a place only after extension question answer pairs connection. During the evaluation process system tried to find a number of possible substitutions for every found path. It takes a long time to evaluate formulated SPARQL queries without any items specified. Evaluation of the number of possible substitutions is a potentially effective metric of the quality assurance. However the number can be influenced by multiple reasons:

- specific areas, such as imported datasets, are covered better and can generate more possible question-answer pairs;
- question simplicity may vary within the evaluation dataset;
- properties directions in the semantic path may overgeneralize possible substitutions.

### Extension evaluation

User experience can be maximized with a supplementation of new question-answer pairs. The subsection evaluates search for a connection between given entities, which is one of core processes of the system. This process will stop when every output entity is linked to any of input entities as described in the section 3.5.

We used evaluation questions and right answers labels to generate a large amount of semantic paths.

Found solutions can be evaluated by characterization of constituent links. The figure 5.2 shows two distributions of found solutions. Left side of the figure shows number of paths per found solution. Large number of the semantic paths in the solution is an indicator of the invalid paths included to the solution. The system automatically rejects solutions with more than 200 semantic paths. The distribution on the right side is a distribution of semantic paths lengths. Semantic path length was initially limited to 4 properties per semantic path to reduce processing time and the amount of invalid paths. That also characterizes dataset structure. An average length between semantically related items is 2.51 properties. A linkage was found in 73.25% cases of input question-answer pairs.

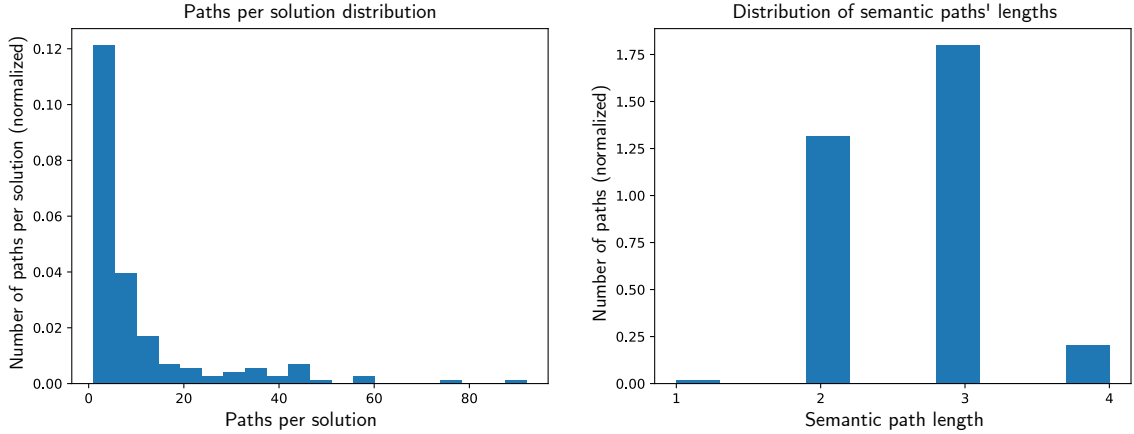


Figure 5.2: Solutions and semantic paths characteristics.

### Answering without a reference

Answering without a reference was initially implemented in an experimental manner to evaluate the system as a general-purpose question answering system.

The following average values were estimated on the set of 228 questions (subset of the evaluation dataset):

$$\overline{Recall} = 0.04 \quad (5.4)$$

$$\overline{Precision} = 0.12 \quad (5.5)$$

$$\overline{F-measure} = 0.06 \quad (5.6)$$

*Note: provided F-measure is global with respect to the total number of questions.*

Obtained measures tell us about low quality of question answering without a reference. F-measure of the QALD participants varies in the range of 0.08 to 0.39 in case of qald-6-train-hybrid and 0.17 to 0.89 in case of qald-6-train-multilingual.

There are multiple reasons which affect the final quality:

- low precision values are related to the scoring process and an adjustment of a score threshold;
- low recall values are related to the scoring process specific to multiple factoid questions;
- evaluation over Wikidata doesn't guarantee the presence of a connection over linked graph;
- approach based on noun phrases implies limitation to specific datatypes.

Another limitation is a zero-length semantic path. There are an example of a question that belongs to the QALD dataset:

What is Batman's real name?

Answer entity Bruce Wayne (Q3645503) has a connection of “said to be the same as” (P460) from the input question entity Batman (Q2695156). The approach tries to find an answer entity at the path between noun phrases entities. For example, between name (Q82799) and Batman (Q2695156). The system can’t answer to that kind of questions due to the system inability to link property of “said to be the same as” (P460) with the “real name” natural language entity.

Current approach of question answering without a reference should be revised in the future versions of the question answering system. Current issues should be resolved following ways:

- the system should link properties with question entities;
- the system should extract items and properties from the set of items related search results;
- we should limit possible directions of links using the syntax tree structure to reduce amount of redundant items during linking with data.

### Processing time

First of all, we should note that systems processing time is related to the graph discovery activities. An example of that sort of activities is a known question-answer pairs extension and an answering without a reference. Typical use case of the system is an answering with a reference. That use case requires two steps:

- matching with data (the average processing time is 0.87s);
- querying of the generated SPARQL query (the average processing time is 0.48s).

Detailed matching with data evaluation will be discussed in the subsection 5.2. The processing time of SPARQL query was estimated by calling recently generated semantic paths. In the real cases embedded entities become different, but the average time can be recognized as more-or-less representative.

Answering with a reference time is acceptable. Complications arise with a graph related activity. An important point is that querying SPARQL is more efficient in terms of time in comparison with a custom implementation of a search over linked data graph. Several steps were done to improve the initial speed of processing.

**Parallel HTTP requests** shown the significant speed improvement. Structural changes of the architecture were required to make parallel requests possible. The system prepares a batch of items or formulated queries. Subsequently the system executes all the queries in a parallel. Matching with data speed was reduced more than ~10x times.

**Dynamic timeout calculation** is another way of time optimization. SPARQL query encounters timeout in case of a general query. The system tries to estimate a processing time while querying on a specific length of a semantic path. Response time of successful queries are used in the process. That improvement of a SPARQL querying reduces processing time more than 3x. Potential improvement may belong to the aggregation of links directions in the semantic path.

Graph-based approach is a part of graph optimization tasks. The whole answering process resides in a balancing between time and quality. At first, adjusting of the system configuration may improve the speed or the quality for specific tasks. Minor processing



speed improvements were obtained with a replace of the Link Grammar Parser by the spaCy NLP.

*Note: Processing time was estimated on the merlin.fit.vutbr.cz server.*

Initialization of the system requires ~10.39s (estimated at 30 runs) that is primarily spent on the spaCy initialization. Processing related to graph manipulations require more time than answering with a reference due to the amount of queries. Finding of a semantic connection takes 35.99s in the average. That search uses question-answer pairs from the prepared dataset. We should note that successful search of a path takes 37.9s, but unsuccessful semantic paths search takes 30.78s. That may be a sign of a demand in a higher timeout values. Semantic paths will be prepared before any user-related evaluations in case of a real application.

Answering without a reference requires ~103.38s to produce an answer. This amount of time isn't appropriate for a question answering purposes. The reason of the result is a variety of possible connection between questions entities. The one of possible solutions is a usage of PODS structures (will be discussed in the section 5.3).

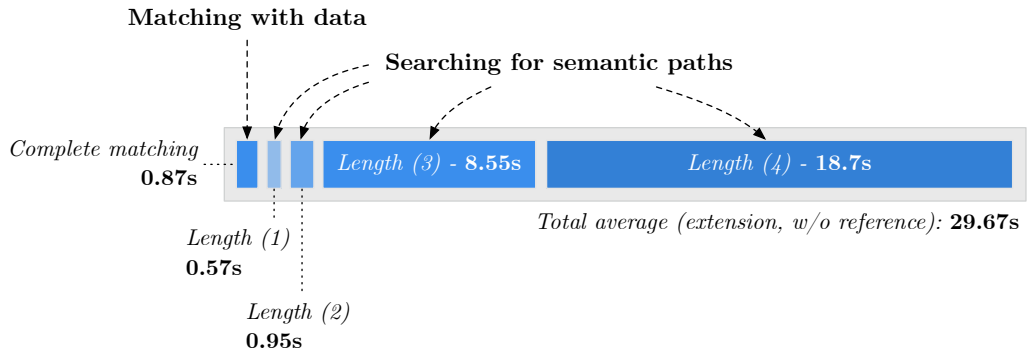


Figure 5.3: Processing time of the processing pipeline's stages.

In the figure 5.3 we can see a comparison of a time on different steps of the processing pipeline. Lengths 1 to 4 in the figure represents an average time spent searching for a semantic path of the corresponding length. We can see an exponential relation between the length of path and the processing time. The growth of the amount of possible connections among items and the growth of possible directions configurations imply the relation.

### Matching with data metrics

Preliminary step of matching with data affects the final processing quality. Parsed noun phrases and successive permutations are involved into the matching. Exponential dependency between number of question entities and a possible connection, which the system should check forcing us to optimization the step of linkage with the data. Merging of entities with intersection inside of search result makes the actual solution possible in an adequate time processing range.

QALD dataset also provides a keyword set for every question sentence. This set was used to evaluate a quality of linking with the data. We should note that our question answering system focuses on linking of noun phrases as a representation of nodes over linked data graph. Provided keywords of the QALD dataset are not obligatory nouns, because different

approach focuses use different sentence parts. Evaluated at 451 question with specified items, our system produces such recall, precision and Fmeasure values:

$$\overline{Recall} = 0.44 \tag{5.7}$$

$$\overline{Precision} = 0.8 \tag{5.8}$$

$$\overline{F-measure} = 0.53 \tag{5.9}$$

F-measure of 0.53 tell us about the relatively high quality of matching process. Matching with the data was an object of optimization, because it affects subsequent processing time.

### Tree matching queries

Process of tree matching was initially covered and discovered in our work related to dialog system. The figure 5.4 provide an example of a syntax tree matching.

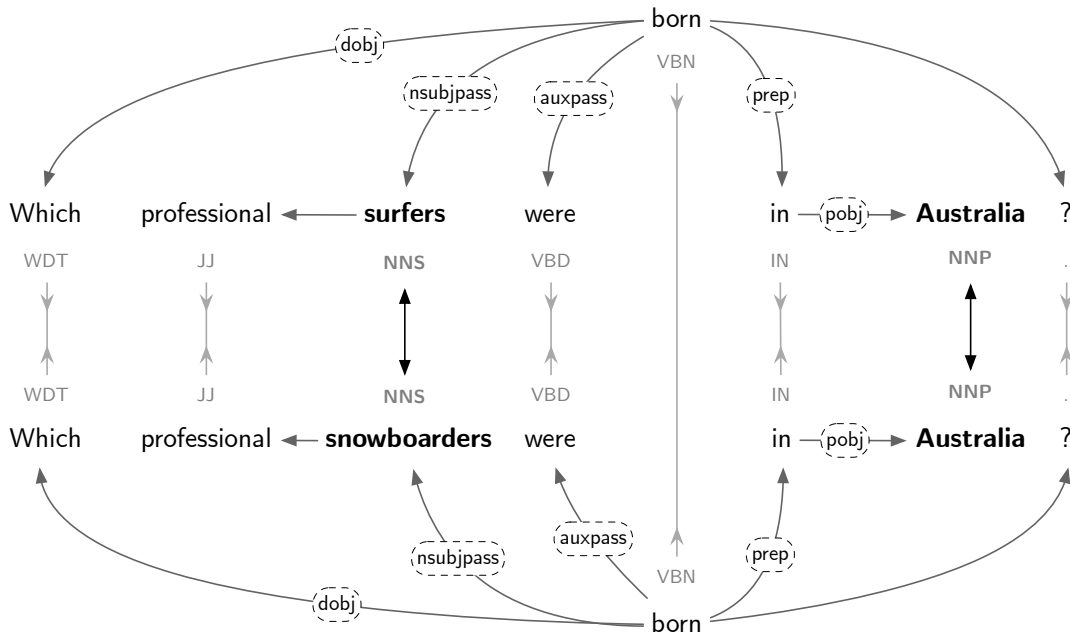


Figure 5.4: Example of a tree matching.

As we can see, system will accept an answer with a higher refinement to produce some answers. Optimal threshold for question matching was calculated during different experiments and trees are recognized as a similar in case of 70% links similarity and entities matching.

Our implementation of the dialog system stores serialized question-answer pairs (the structure is covered at the figure 3.3) in a binary file and wasn't focused on a high-volume question answering. That can be easily optimized by usage of traditional database system.

Precondition and postcondition steps are related to question/answer acceptance. As long as postcondition is only related to complex answer structures obtained by end-users it

Table 5.1: Question type classification measures.

Question type	Processed	Processed (%)	Precision	Recall
<i>single resource</i>	253	40.03%	0.78	0.84
<i>multiple resources</i>	191	30.22%	0.77	0.80
<i>number</i>	81	12.82%	0.74	0.80
<i>date</i>	28	4.43%	0.80	0.86
<i>boolean</i>	55	8.7%	0.81	0.78

wasn't evaluated yet. Precondition search has no difference with a naive question answering without a reference. Final percentage of precondition persistence was obtained during answering evaluation (38%). That percentage of questions contains at least one link between input sentence entities. We can recommend to limit the semantic connection between question items to one property, as long as longer connections can significantly reduce question matching rate.

### Classification metric

The system searches for syntax tree substructures while classifying question types. Quality of question type classification affects the final result in case of answering without a reference. In case of invalid question classification the system will score found semantic paths using invalid heuristic. Prepared dataset contains a question types from QALD datasets. For every type of question we calculated questions classification (see table 5.1).

## 5.3 Comparison with existing solutions

### Treo

Treo is a Semantic Search and Question Answering System for Databases. Treo is designed to cope with the Big Data vision of handling very heterogeneous databases. The system focuses on scenarios, at which it becomes unfeasible for data consumers to understand the representation of the data in order to query the database [31].

Processing starts with consists in determining the key entities in the user query and mapping the entities in the query to entities on datasets. The first step of processing is similar in most approaches. The mapping from the natural language terms representing the entities to the URIs representing these entities in the datasets is done through entity search step. The URIs define the pivot entities in the datasets, which are the entry points for the semantic search process.

At the next step the user natural language query is pre-processed into a Partial Ordered Dependency Structure (PODS). PODS is a format which is close to the triple-like (subject, predicate and object) structure of RDF. The construction of the PODS demands the previous entity recognition step. In opposition to our system Treo extracts triple relation from the question tree structure.

Taking as inputs the pivot entities URIs and the PODS query representation, the semantic matching process starts by fetching all the relations associated with the top ranked

pivot entity. Starting from the pivot entity, the labels of each relation associated with the pivot node have their semantic relatedness measured against the next term in the PODS representation of the query. The relations with the highest proximity measures define the neighboring nodes. These nodes will be explored in the search process. The search algorithm then navigates to the nodes with high relatedness values, where the same process happens for the next query term. The search process continues until the end of the query is reached, working as a spreading activation search over the RDF graph. Activation function (the threshold to determine the further node exploration process) is defined by a semantic relatedness measure [31].

Treo algorithm is optimized to complexity of a query because it moves through a pivot entities the same way as syntax tree structure connects entities. Comparison of initial item properties and linked items has a beneficial effect to the process, without any limitation to noun-only entities. Triple PODS implementation also defines a unified scoring while extending items. Linked data graph may contain directed links, which are not used while extending pivot elements in single direction [30]. The approach faces the issues with the granularity of data, so multiple links should be used in case of granular property. Semantic proximity measure adopts the approach to granularity of a dataset.

### **Top-k approach**

Top-k approach is also a graph related approach, IR concepts are adapted to support an imprecise matching that incorporates syntactic and semantic similarities. As a result, the user does not need to know the labels of the data elements when doing keyword search.

Traditional approaches interpret keywords as neighbors of answers. Proposed in the [55] approach interpret keywords as elements of structured queries. Instead of presenting the top-k answers, which might actually belong to many distinct queries, top-k approach let the user select one of the top-k queries to retrieve all its answers. Thus, the keyword search process contains an additional step, namely the presentation of structured queries. Refinement can be made more precisely on the structured query than on the keyword query [38].

Main technical contribution of the approach is a novel algorithm for the computation of the top-k subgraphs. Traditionally keywords are exclusively mapped to vertices. In order to connect the vertices corresponding to the keywords, top-k algorithm aim at computing tree-shaped candidate networks or answer trees. Since keywords do not necessarily correspond to answers exclusively in our approach, they might also be mapped to edges. As a consequence, substructures connecting keyword elements are not restricted to trees, but can be graphs in general. Therefore, algorithms as applied for tree-exploration such as breadth-first search, backward search or bidirectional search are not sufficient.

The top-k approach constructs possible queries over linked data instead of searching of an item. Our answering without a reference approach has a relation to searching for an item over linked data graph approaches. Reference sentence in our approach makes our approach strict, but efficient.

## Chapter 6

# Conclusions and potential extensions

We have created a question answering system which uses and improves a graph-based approach. The system was evaluated using standard metrics in this area of research and the adapted dataset, it was tested in real use cases. This solution is designed as a dataset independent tool with abilities of extension using natural language, not as a query construction tool. We can see benefits and potential advantages of graph discovery at every step of the answering pipeline. Graph comparison allows to match input queries retaining the context, apply relevant graph path to input data and construct a natural language answer using reference. The question answering system focuses on the answering with a reference question-answer pair provided by the user but also implements regular entity search method in order to the compatibility with similar systems. Our schema-agnostic system uses general principles of linked data and is able to fit custom knowledge representations.

Drawback of the graph-based approach reveals itself in case of non-factoid questions. Being initially designed for factoid questions our system isn't an eligible option in case of aggregation and non-standard data type queries. Focus on structured data limits the system when it comes to the tasks that require hybrid techniques.

Developed system was evaluated using standard metrics. Answering quality and linking rate 73.25% of the initial approach are acceptable, while an attempt to adapt the approach to standard metrics have been unsuccessful with answering without reference F-measure of 0.03. Obtained answering and extension time is acceptable for artificial querying, but not comparable with real use cases. Processing time is counterbalanced by the final precision. While working with heterogeneous and inconsistent data is a possible way to find balance between precision and response time. This work binds sentence structures with the connection over linked data graph, but there are a lot of challenges and appropriate techniques in the area of question answering over linked data which can significantly improve answering process when correctly combined.

As for practical usage of question answering system, it is possible to create a reference question-answer pair, evaluate quality of the found linkage and query similarly structured question with a reply. Designed as a multi-user system, it is able to redirect an unknown question between users during work in a server mode. Final solution is extensible by means of natural language and abstracts data consumers from the data structure.

Modular architecture of the system allows to reuse components as part of any other Python conversational agent to access prominent knowledge bases. Highly optimized match-

ing module that introduces structural permutations demonstrates F-measure of 0.53 and an average processing time of 0.74 seconds. Another beneficial influence of the modularity is a possibility to integrate custom data representations. In case of custom linked data implementation this system can lower the barrier for extensions of a domain independent system.

Potential extension belongs to these categories:

- optimization of the processing time;
- answering without a reference improvements;
- general improvements that affects the proposed approach.

*Processing time optimization* is essential for practical usage over large-scale datasets. Graph-based systems are sensitive to the granularity of the provided dataset and link qualities. General domain usage over prominent knowledge bases and especially over Wikidata is also possible and should belong to processing time improvements by usage of local knowledge base instances and endpoints.

*Answering without a reference* extension may belong to the scoring heuristics, for example in spreading activation guided by a measure of semantic relatedness. Self-sufficient semantic paths are not an eligible option for a complete inference of the answer. Future research may comprise matching words featuring a part-of-speech other than noun with item properties and semantic paths scoring.

# Bibliography

- [1] Apache Lucene Core. <https://lucene.apache.org/core/> [Online; accessed 20-May-2017].
- [2] Apache UIMA. <https://uima.apache.org> [Online; accessed 20-May-2017].
- [3] Comparison of RDFJS libraries. [https://www.w3.org/community/rdfjs/wiki/Comparison\\_of\\_RDFJS\\_libraries](https://www.w3.org/community/rdfjs/wiki/Comparison_of_RDFJS_libraries) [Online; accessed 20-May-2017].
- [4] GitHub - linkeddata/rdfliib.js: Linked Data API for JavaScript. <https://github.com/linkeddata/rdfliib.js/> [Online; accessed 20-May-2017].
- [5] How spaCy Works | Explosion AI Blog. <https://explosion.ai/blog/how-spacy-works> [Online; accessed 20-May-2017].
- [6] NLTK 3.0 documentation. <http://www.nltk.org> [Online; accessed 11-Jan-2017].
- [7] Rdfstore-js. <https://www.w3.org/2001/sw/wiki/Rdfstore-js> [Online; accessed 20-May-2017].
- [8] Requests: HTTP for Humans. <http://docs.python-requests.org/en/master/> [Online; accessed 20-May-2017].
- [9] Stanford CoreNLP – a suite of core NLP tools. <http://stanfordnlp.github.io/CoreNLP/> [Online; accessed 11-Jan-2017].
- [10] TextBlob: Simplified Text Processing. <https://textblob.readthedocs.io/en/dev/> [Online; accessed 11-Jan-2017].
- [11] WordNet: A Lexical Database for English. <https://wordnet.princeton.edu> [Online; accessed 11-Jan-2017].
- [12] Baudiš, P.: Systems and Approaches for Question Answering. Technical report. Czech Technical University. 2015.
- [13] Baudiš, P.: YodaQA: A Modular Question Answering System Pipeline. Technical report. Czech Technical University. 2015.
- [14] Berners-Lee, T.: Linked Data. 2006. <http://www.w3.org/DesignIssues/LinkedData.html> [Online; accessed 11-Jan-2017].
- [15] Bernstein, A.; Kaufmann, E.; Kaiser, C.: Querying the semantic web with ginseng: A guided input natural language search engine. In *15th Workshop on Information Technologies and Systems, Las Vegas, NV*. 2005. pp. 112–126.

- [16] BI Intelligence: Messaging apps are now bigger than social networks. September 2016. [Online; posted 20-September-2016]  
<http://www.businessinsider.com/the-messaging-app-report-2015-11>.
- [17] Bird, S.; Klein, E.; Loper, E.: *Natural Language Processing with Python*. O'Reilly Media, Inc.. first edition. 2009. ISBN 0596516495, 9780596516499.
- [18] Birger, M.: Dialog system for Human-Robot communication. 2015.
- [19] Bizer, C.; Heath, T.; Idehen, K.; et al.: Linked Data on the Web (LDOW2008). In *Proceedings of the 17th International Conference on World Wide Web. WWW '08*. New York, NY, USA: ACM. 2008. ISBN 978-1-60558-085-2. pp. 1265–1266.  
doi:10.1145/1367497.1367760.  
Retrieved from: <http://doi.acm.org/10.1145/1367497.1367760>
- [20] Bizer, C.; Lehmann, J.; Kobilarov, G.; et al.: {DBpedia} - A crystallization point for the Web of Data. *Web Semantics: Science, Services and Agents on the World Wide Web*. vol. 7, no. 3. 2009: pp. 154 – 165. ISSN 1570-8268.  
doi:http://dx.doi.org/10.1016/j.websem.2009.07.002. the Web of Data.  
Retrieved from:  
<http://www.sciencedirect.com/science/article/pii/S1570826809000225>
- [21] Bouziane, A.; Bouchiha, D.; Doumi, N.; et al.: Question Answering Systems: Survey and Trends. *Procedia Computer Science*. vol. 73. 2015: pp. 366 – 375. ISSN 1877-0509. doi:http://dx.doi.org/10.1016/j.procs.2015.12.005. international Conference on Advanced Wireless Information and Communication Technologies (AWICT 2015).  
Retrieved from:  
<http://www.sciencedirect.com/science/article/pii/S1877050915034663>
- [22] Cimiano, P.; Lopez, V.; Unger, C.; et al.: *Multilingual Question Answering over Linked Data (QALD-3): Lab Overview*. Berlin, Heidelberg: Springer Berlin Heidelberg. 2013. ISBN 978-3-642-40802-1. pp. 321–332.  
doi:10.1007/978-3-642-40802-1\_30.  
Retrieved from: [http://dx.doi.org/10.1007/978-3-642-40802-1\\_30](http://dx.doi.org/10.1007/978-3-642-40802-1_30)
- [23] Daiber, J.; Jakob, M.; Hokamp, C.; et al.: Improving Efficiency and Accuracy in Multilingual Entity Extraction. In *Proceedings of the 9th International Conference on Semantic Systems (I-Semantics)*. 2013.
- [24] Damljanovic, D.; Agatonovic, M.; Cunningham, H.: *Natural Language Interfaces to Ontologies: Combining Syntactic Analysis and Ontology-Based Lookup through the User Interaction*. Berlin, Heidelberg: Springer Berlin Heidelberg. 2010. ISBN 978-3-642-13486-9. pp. 106–120. doi:10.1007/978-3-642-13486-9\_8.  
Retrieved from: [http://dx.doi.org/10.1007/978-3-642-13486-9\\_8](http://dx.doi.org/10.1007/978-3-642-13486-9_8)
- [25] DBpedia Association: DBpedia About. 2016. <http://wiki.dbpedia.org/about> [Online; accessed 11-Jan-2017].
- [26] Delparish, M.: Toward development of human-computer interaction using natural language processing. *Indian Journal of Fundamental and Applied Life Sciences*. vol. 5, no. S1. 2015.



- [27] Erxleben, F.; Günther, M.; Krötzsch, M.; et al.: *Introducing Wikidata to the Linked Data Web*. Cham: Springer International Publishing. 2014. ISBN 978-3-319-11964-9. pp. 50–65. doi:10.1007/978-3-319-11964-9\_4.  
Retrieved from: [http://dx.doi.org/10.1007/978-3-319-11964-9\\_4](http://dx.doi.org/10.1007/978-3-319-11964-9_4)
- [28] Ferrucci, D.; Brown, E.; Chu-Carroll, J.; et al.: Building Watson: An Overview of the DeepQA Project. *AI Magazine*. vol. 31, no. 3. 2010.  
<http://www.aaai.org/ojs/index.php/aimagazine/article/view/2303/0>.
- [29] Ferrucci, D.; Nyberg, E.; Allan, J.; et al.: IBM Research Report: Towards the Open Advancement of Question Answering Systems. Technical report. IBM Research Division. 2009. <http://domino.watson.ibm.com/library/CyberDig.nsf/papers/D12791EAA13BB952852575A1004A055C>.
- [30] Freitas, A.; Curry, E.; Oliveira, J. G.; et al.: Querying Heterogeneous Datasets on the Linked Data Web: Challenges, Approaches, and Trends. *IEEE Internet Computing*. vol. 16, no. 1. Jan 2012: pp. 24–33. ISSN 1089-7801. doi:10.1109/MIC.2011.141.
- [31] Freitas, A.; de Faria, F. F.; O’Riain, S.; et al.: Answering Natural Language Queries over Linked Data Graphs: A Distributional Semantics Approach. In *Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR ’13. New York, NY, USA: ACM. 2013. ISBN 978-1-4503-2034-4. pp. 1107–1108. doi:10.1145/2484028.2484209.  
Retrieved from: <http://doi.acm.org/10.1145/2484028.2484209>
- [32] He, S.; Liu, S.; Chen, Y.; et al.: CASIA@QALD-3: A Question Answering System over Linked Data. Technical report. Chinese Academy of Sciences. 2013.  
<https://pdfs.semanticscholar.org/3490/d0610484b676b0270a3136e3ca819ea22d79.pdf>.
- [33] Hendrix, G. G.; Sacerdoti, E. D.; Sagalowicz, D.; et al.: Developing a Natural Language Interface to Complex Data. *ACM Trans. Database Syst.* vol. 3, no. 2. June 1978: pp. 105–147. ISSN 0362-5915. doi:10.1145/320251.320253.  
Retrieved from: <http://doi.acm.org/10.1145/320251.320253>
- [34] Hoffart, J.; Suchanek, F. M.; Berberich, K.; et al.: YAGO2: Exploring and Querying World Knowledge in Time, Space, Context, and Many Languages. In *Proceedings of the 20th International Conference Companion on World Wide Web*. WWW ’11. New York, NY, USA: ACM. 2011. ISBN 978-1-4503-0637-9. pp. 229–232. doi:10.1145/1963192.1963296.  
Retrieved from: <http://doi.acm.org/10.1145/1963192.1963296>
- [35] Huberman, B. A.; Adamic, L. A.: Internet: Growth dynamics of the World-Wide Web. *Nature*. vol. 401, no. 131. 1999.
- [36] Kalaivani, S.; Duraiswamy, K.: Comparison of Question Answering Systems Based on Ontology and Semantic Web in Different Environment. *Journal of Computer Science*. vol. 8, no. 9. 2012: pp. 1407 – 1413. doi:10.3844/jcssp.2012.1407.1413.  
Retrieved from: <http://thescipub.com/html/10.3844/jcssp.2012.1407.1413>
- [37] Lehmann, J.; Isele, R.; Jakob, M.; et al.: DBpedia – A Large-scale, Multilingual Knowledge Base Extracted from Wikipedia. *Semantic Web – Interoperability, Usability, Applicability*. vol. 1, no. 5. 2012.

<http://www.semantic-web-journal.net/content/dbpedia-large-scale-multilingual-knowledge-base-extracted-wikipedia-0>.

- [38] Liu, Q.; Gao, Y.; Chen, G.; et al.: Answering why-not and why questions on reverse top-k queries. *The VLDB Journal*. vol. 25, no. 6. 2016: pp. 867–892. ISSN 0949-877X. doi:10.1007/s00778-016-0443-4.  
Retrieved from: <http://dx.doi.org/10.1007/s00778-016-0443-4>
- [39] Lopez, V.; Motta, E.; Uren, V.: *PowerAqua: Fishing the Semantic Web*. Berlin, Heidelberg: Springer Berlin Heidelberg. 2006. ISBN 978-3-540-34545-9. pp. 393–410. doi:10.1007/11762256\_30.  
Retrieved from: [http://dx.doi.org/10.1007/11762256\\_30](http://dx.doi.org/10.1007/11762256_30)
- [40] Lopez, V.; Unger, C.; Cimiano, P.; et al.: Evaluating question answering over linked data. *Web Semantics: Science, Services and Agents on the World Wide Web*. vol. 21. 2013: pp. 3 – 13. ISSN 1570-8268.  
doi:<http://dx.doi.org/10.1016/j.websem.2013.05.006>. special Issue on Evaluation of Semantic Technologies.  
Retrieved from:  
<http://www.sciencedirect.com/science/article/pii/S157082681300022X>
- [41] Mahdisoltani, F.; Biega, J.; Suchanek, F. M.; et al.: YAGO3: A Knowledge Base from Multilingual Wikipedias.
- [42] Manning, C. D.; Raghavan, P.; Schütze, H.: *Introduction to Information Retrieval*. Cambridge University Press. first edition. 2008. ISBN 0521865719.
- [43] Manning, C. D.; Surdeanu, M.; Bauer, J.; et al.: The Stanford CoreNLP Natural Language Processing Toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*. 2014. pp. 55–60.  
Retrieved from: <http://www.aclweb.org/anthology/P/P14/P14-5010>
- [44] Miller, G. A.: WordNet: A Lexical Database for English. *Commun. ACM*. vol. 38, no. 11. November 1995: pp. 39–41. ISSN 0001-0782. doi:10.1145/219717.219748.  
Retrieved from: <http://doi.acm.org/10.1145/219717.219748>
- [45] Paulheim, H.: Knowledge Graph Refinement: A Survey of Approaches and Evaluation Methods. *Semantic Web – Interoperability, Usability, Applicability*. vol. 1, no. 0. 2016. <http://www.semantic-web-journal.net/content/knowledge-graph-refinement-survey-approaches-and-evaluation-methods>.
- [46] RDF Data Access Working Group: SPARQL Query Language for RDF. 2013. <https://www.w3.org/TR/rdf-sparql-query/> [Online; accessed 11-Jan-2017].
- [47] RDF Working Group: Resource Description Framework (RDF). 2014. <https://www.w3.org/RDF/> [Online; accessed 11-Jan-2017].
- [48] Rizzo, G.; Troncy, R.: NERD: evaluating named entity recognition tools in the web of data. In *ISWC 2011, Workshop on Web Scale Knowledge Extraction (WEKEX'11), October 23-27, 2011, Bonn, Germany*. Bonn, ALLEMAGNE. 10 2011.  
Retrieved from: <http://www.eurecom.fr/publication/3517>

- [49] Shekarpour, S.; Lukovnikov, D.; Kumar, A. J.; et al.: Question Answering on Linked Data: Challenges and Future Directions. *CoRR*. vol. abs/1601.03541. 2016. Retrieved from: <http://arxiv.org/abs/1601.03541>
- [50] Singhal, A.: Introducing the Knowledge Graph: things, not strings. 2012. <https://googleblog.blogspot.cz/2012/05/introducing-knowledge-graph-things-not.html> [Online; accessed 11-Jan-2017].
- [51] Sleator, D. D. K.; Temperley, D.: Parsing English with a Link Grammar. Technical report. School of Computer Science, Carnegie Mellon University. Pittsburgh, PA, USA. 1991. <http://www.cs.cmu.edu/afs/cs.cmu.edu/project/link/pub/www/papers/ps/tr91-196.pdf>.
- [52] Socher, R.; Lin, C. C.; Ng, A. Y.; et al.: Parsing Natural Scenes and Natural Language with Recursive Neural Networks. In *Proceedings of the 26th International Conference on Machine Learning (ICML)*. 2011.
- [53] Suchanek, F.; Kasneci, G.; Weikum, G.: YAGO: A Core of Semantic Knowledge - Unifying WordNet and Wikipedia. In *16th International World Wide Web Conference (WWW 2007)*, edited by C. L. Williamson; M. E. Zurko; P. J. Patel-Schneider, Peter F. Shenoy. Banff, Canada: ACM. 2007. ISBN 978-1-59593-654-7. pp. 697–706. doi:<http://doi.acm.org/10.1145/1242572.1242667>.
- [54] The DeepQA Research Team: DeepQA. [http://researcher.watson.ibm.com/researcher/view\\_group\\_subpage.php?id=2159](http://researcher.watson.ibm.com/researcher/view_group_subpage.php?id=2159) [Online; accessed 11-Jan-2017].
- [55] Tran, T.; Wang, H.; Rudolph, S.; et al.: Top-k Exploration of Query Candidates for Efficient Keyword Search on Graph-Shaped (RDF) Data. In *2009 IEEE 25th International Conference on Data Engineering*. March 2009. ISSN 1063-6382. pp. 405–416. doi:10.1109/ICDE.2009.119.
- [56] Trotman, A.; Geva, S.; Kamps, J.: Report on the SIGIR 2007 Workshop on Focused Retrieval. *SIGIR Forum*. vol. 41, no. 2. December 2007: pp. 97–103. ISSN 0163-5840. doi:10.1145/1328964.1328981. Retrieved from: <http://doi.acm.org/10.1145/1328964.1328981>
- [57] Unger, C.; Freitas, A.; Cimiano, P.: *An Introduction to Question Answering over Linked Data*. Cham: Springer International Publishing. 2014. ISBN 978-3-319-10587-1. pp. 100–140. doi:10.1007/978-3-319-10587-1\_2. Retrieved from: [http://dx.doi.org/10.1007/978-3-319-10587-1\\_2](http://dx.doi.org/10.1007/978-3-319-10587-1_2)
- [58] Unger, C.; Ngomo, A.-C. N.; Cabrio, E.: *6th Open Challenge on Question Answering over Linked Data (QALD-6)*. Cham: Springer International Publishing. 2016. ISBN 978-3-319-46565-4. pp. 171–177. doi:10.1007/978-3-319-46565-4\_13. Retrieved from: [http://dx.doi.org/10.1007/978-3-319-46565-4\\_13](http://dx.doi.org/10.1007/978-3-319-46565-4_13)
- [59] Valencia-García, R.; García-Sánchez, F.: {NATURAL} {LANGUAGE} {PROCESSING} and Human–Computer Interaction. *Computer Standards & Interfaces*. vol. 35, no. 5. 2013: pp. 415 – 416. ISSN 0920-5489. doi:<http://dx.doi.org/10.1016/j.csi.2013.03.001>.

Retrieved from:

<http://www.sciencedirect.com/science/article/pii/S0920548913000196>

- [60] Vanessa Lopez, M. S., Enrico Motta; Fernandez, M.: PowerAqua: A Multi-Ontology Based Question Answering System – v1. Technical report. Knowledge Media Institute, The Open University. Milton Keynes, United Kingdom. 2007.  
[http://technologies.kmi.open.ac.uk/aqualog/okdeliverable/OK\\_D8.4\\_PowerAqua\\_vfinal.pdf](http://technologies.kmi.open.ac.uk/aqualog/okdeliverable/OK_D8.4_PowerAqua_vfinal.pdf).
- [61] Vrandečić, D.; Krötzsch, M.: Wikidata: A Free Collaborative Knowledge Base. *Communications of the ACM*. vol. 57, no. 10. 2014: pp. 78 – 85.  
Retrieved from:  
<http://cacm.acm.org/magazines/2014/10/178785-wikidata/fulltext>
- [62] Wikimedia Foundation, Inc.: Wikidata. 2016.  
[https://www.wikidata.org/wiki/Wikidata:Main\\_Page](https://www.wikidata.org/wiki/Wikidata:Main_Page) [Online; accessed 11-Jan-2017].
- [63] Xu, K.; Feng, Y.; Zhao, D.: Xser@QALD-4: Answering Natural Language Questions via Phrasal Semantic Parsing. Technical report. Peking University. 2014.  
<http://ceur-ws.org/Vol-1180/CLEF2014wn-QA-XuEt2014.pdf>.

# Appendix A

## CD Content

```
.
|-- poster.pdf          poster
|-- sources
|  |-- LICENSE.txt     MIT license
|  |-- README.md       readme and manual
|  |-- config.ini      default configuration
|  |-- evaluation
|  |  |-- dataset.json prepared evaluation dataset
|  |  |-- evaluate.py  evaluation of the
|  |  |-- generate.py  generation of the dataset
|  |  |-- processing.py generate figures
|  |  +-- ...         helper scripts
|  |-- init.sh         environment setup script
|  |-- qas            question answering system sources
|  |  |-- __init__.py
|  |  |-- __main__.py
|  |  |-- core.py
|  |  |-- dbpedia.py
|  |  |-- graph.py
|  |  |-- items.py
|  |  |-- link_grammar.py
|  |  |-- logs.py
|  |  |-- noun_phrase.py
|  |  |-- scenarios.py
|  |  |-- sentence.py
|  |  +-- wikidata.py
|  |-- requirements.txt dependencies list
|  |-- server         server wrapper
|  |  +-- ...
|  +-- setup.py      installation scripts
|-- thesis          tech. report LaTeX sources
|  +-- ...
+-- thesis.pdf      tech. report
```

# Appendix B

## Manual

### B.1 Virtual environment installation

Please, make sure, that you have a `python3` (3.5.2) and `pip3` and/or `easy_install-3.x` installed. Also `curl` utility is needed.

```
sudo apt-get install python3-setuptools python3.6-dev python3-pip
```

It is possible to create a virtual environment with python dependencies. Make sure, that you have `virtualenv`. To check that `virtualenv` is installed:

```
which virtualenv
```

You should see a path to the executable. If it is available, you can run the initialization script:

```
. ./init.sh
```

`init.sh` file creates a virtual environment, install needed dependencies and download WordNet and spaCy corporas.

After, you can run the system:

```
qa_system --help
```

To leave the virtual environment:

```
deactivate
```

To activate the environment again:

```
source env/bin/activate
```

To leave and delete the virtual environment:

```
. ./clean.sh
```

## B.2 External dependencies

All dependencies will be automatically installed in case of the environment setup. This dependencies required only in case of a custom installation.

The QAS uses spaCy for natural language processing. You can install spaCy using pip:

```
pip install -U spacy
```

SpaCy requires an english model (~1GB) as well:

```
python -m spacy download en
```

The second required dependency is the NLTK:

```
echo "import nltk; nltk.download('wordnet')" > download_wordnet.py
```

For WebSocket wrapper you also need a Flask webserver:

```
pip3 install flask
```

## B.3 Getting started

Generating the evaluation dataset:

```
cd evaluation
python3 generate.py qald/ | tee dataset.test.json
```

Running the evaluation script:

```
python3 evaluate.py dataset.test.json 2> processing.log
```

Generating distribution graphs:

```
python3 processing.py
```

To manually run the system under the environment:

```
qa_system -q "In what city is the Heineken brewery?"
```

Please, provide a question-answer pair to find a semantic path:

```
qa_system -q "In what city is the Heineken brewery?" -a "Amsterdam"
```

## Appendix C

# Evaluation dataset

Listing C.1: Fragment of the evaluation dataset.

---

```
1 {
2   "questions": [
3     {
4       "aggregation": "false",
5       "answers": [
6         {
7           "answertype": "resource",
8           "dbpedia": "http://dbpedia.org/resource/Barack_Obama",
9           "wikidata": "http://www.wikidata.org/entity/Q76"
10        }
11      ],
12      "answertype": "resource",
13      "dataset": "qald-6-test-hybrid",
14      "items": [],
15      "keywords": [],
16      "string": "Who is the president by whom 20 million Americans had
17                gained health insurance?"
18    }, {
19      "aggregation": "true",
20      "answers": [
21        {
22          "answertype": "resource",
23          "dbpedia": "http://dbpedia.org/resource/Indian_Railways",
24          "wikidata": "http://www.wikidata.org/entity/Q819425"
25        }
26      ],
27      "answertype": "resource",
28      "dataset": "qald-6-test-multilingual",
29      "items": [
30        "Indian company",
31        "the most employees"
32      ],
33      "keywords": [
```



```

34     "Indian company",
35     "the most employees"
36 ],
37 "string": "Which Indian company has the most employees?"
38 }, {
39     "aggregation": "false",
40     "answers": [
41         {
42             "answertype": "number",
43             "value": "6"
44         }
45     ],
46     "answertype": "number",
47     "dataset": "qald-6-test-hybrid",
48     "items": [],
49     "keywords": [],
50     "string": "How many awards achieved the creator of the world's
51             most famous equation?"
52 }, {
53     "aggregation": "false",
54     "answers": [
55         {
56             "answertype": "resource",
57             "dbpedia": "http://dbpedia.org/resource/Rick_Husband",
58             "wikidata": "http://www.wikidata.org/entity/Q346671"
59         },
60         {
61             "answertype": "resource",
62             "dbpedia": "http://dbpedia.org/resource/William_C._McCool",
63             "wikidata": "http://www.wikidata.org/entity/Q334463"
64         },
65         {
66             "answertype": "resource",
67             "dbpedia": "http://dbpedia.org/resource/Michael_P._Anderson",
68             "wikidata": "http://www.wikidata.org/entity/Q318542"
69         },
70         {
71             "answertype": "resource",
72             "dbpedia": "http://dbpedia.org/resource/Ilan_Ramon",
73             "wikidata": "http://www.wikidata.org/entity/Q219569"
74         },
75         {
76             "answertype": "resource",
77             "dbpedia": "http://dbpedia.org/resource/Kalpana_Chawla",
78             "wikidata": "http://www.wikidata.org/entity/Q237879"
79         },
80         {
81             "answertype": "resource",

```

```

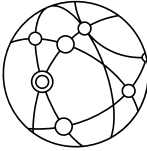
82         "dbpedia": "http://dbpedia.org/resource/David_M._Brown",
83         "wikidata": "http://www.wikidata.org/entity/Q350317"
84     }
85 ],
86     "answertype": "resource",
87     "dataset": "qald-6-test-hybrid",
88     "items": [],
89     "keywords": [],
90     "string": "Which astronauts died at the Columbia disaster
91               also known as STS-107?"
92 }, {
93     "aggregation": "false",
94     "answers": [true],
95     "answertype": "boolean",
96     "dataset": "qald-6-train-hybrid",
97     "items": [],
98     "keywords": [],
99     "string": "Did Napoleon's first wife die in France?"
100 }, {
101     "aggregation": false,
102     "answers": [
103         {
104             "answertype": "resource",
105             "dbpedia": "http://dbpedia.org/resource/Marcus_Adoro",
106             "wikidata": "http://www.wikidata.org/entity/Q3545252"
107         }
108     ],
109     "answertype": "resource",
110     "dataset": "qald-7-train-multilingual",
111     "items": [
112         "professional surfer",
113         "born",
114         "Philippines"
115     ],
116     "keywords": [
117         "professional surfer",
118         "born",
119         "Philippines"
120     ],
121     "string": "Which professional surfers were born on the Philippines?"
122 }
123 ]
124 }

```

---

# Appendix D

## Poster



# Question Answering System

**MOTIVATION**

Nowadays **natural language interfaces** become popular.

We can see the exponential growth of the **Semantic Web**.

It is important to note that companies accumulate **custom data**.

**ASSUMPTION**

💡  
*"if a known question has a known answer, we can recognize an input question as similar and produce an answer with the same semantic links over the knowledge base"*


**APPROACH**

**Graph-based** system explores provided graph to yield and reuse semantic connections.

**Schema-agnostic approach** allows to hide the underlying schema from users.


**USE CASES**

Answering with a reference:



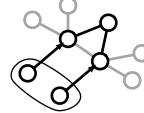
The system substitutes answers using syntax trees.

Extending known question-answer pairs:



The system can be extended with the use of natural language.




Answering without a reference:




Methods of answering without a reference were implemented.

**EVALUATION**

Prominent **linked datasets** were used in the project. The system was evaluated using **standard metrics**.



**CREDITS**




BRNO UNIVERSITY OF TECHNOLOGY

FACULTY OF INFORMATION TECHNOLOGY

AUTHOR: **Mark Birger**  
SUPERVISOR: **Doc. RNDr. PAVEL SMRŽ, Ph.D.**

**DISTRIBUTION**

Source code is available at the Github under the MIT license.



Icons by: The Noun Project, Python Software Foundation, W3C, Wikidata, DBpedia. (CC BY 3.0)

Figure D.1: Research illustration.