Semantic Question Answering Over Knowledge Graphs: Pitfalls and Pearls

Dissertation
zur
Erlangung des Doktorgrades (Dr. rer. nat.)
der
Mathematisch-Naturwissenschaftlichen Fakultät
der
Rheinischen Friedrich-Wilhelms-Universität Bonn

von
Hamid Zafartavanaelmi
aus
Neyshaboor, Iran

Bonn, 08.2020

Dieser Forschungsbericht wurde als Dissertation von der Mathematisch-Naturwissenschaftlichen Fakultät der Universität Bonn angenommen und ist auf dem Hochschulschriftenserver der ULB Bonn https://nbn-resolving.org/urn:nbn:de:hbz:5-61546 elektronisch publiziert.

1. Gutachter:	Prof. Dr. Jens Lehmann
2. Gutachter:	Prof. Dr. Axel-Cyrille Ngonga Ngomo
Tag der Promotion:	03.03.2021
Erscheinungsjahr:	2021

Abstract

Nowadays, the Web provides an infrastructure to share all kinds of information which are easily accessible to humans around the world. Furthermore, the amount of information is growing rapidly and requires computing machines to process, comprehend, and extract useful information tailored for the end-users. The Semantic Web and semantic technologies play a prominent role to enable knowledge representation and reasoning for these computational processes. Semantic technologies such as ontologies and knowledge graphs are being used in various application domains, including data governance, knowledge management, chatbots, biology, etc., which aim at providing proper infrastructure to analyze the knowledge and reasoning for the computers. Semantic Question Answering systems are among the most desired platforms in recent years that facilitate access to information in knowledge graphs. They provide a natural language interface that permits the users to ask their questions posed in a natural language, without any understanding of the underlying technologies. We thus study question answering systems over knowledge graphs which aim to map an input question in natural language into a formal query, intending to retrieve a concise answer from the knowledge graph. This is a highly challenging task due to the intrinsic complexity of the natural language, such that the resulting query does not always accurately subserve the user intent, particularly, for more complex and less common questions.

In this thesis, we explore semantic question answering systems in a modular manner in order to discover the bottlenecks and mitigate the challenges in each part independently. Therefore, we focus on the individual modules and propose two innovative models: First, a reinforcement learning-based approach to parse the input question using distant labels, and second, an algorithm that generates the candidate formal queries based on a set of linked entities and relations. The latter additionally uses a neural network based model to rank the candidate queries by exploiting the structural similarity of the input question and the candidate queries. Through extensive empirical studies, we demonstrate that our proposed models perform well on three commonly used question answering datasets and increase the overall performance of the question answering system. In addition, we design an interactive question answering system that solicits users for their feedback with the aim of guiding the system toward seizing the rectified semantic query, while striving to keep user satisfaction into account. Our oracle evaluation indicates that even a small number of user interactions can lead to a significant improvement in the performance of semantic question answering systems. Moreover, we conduct a user study to evaluate the performance of our system in interactive scenarios. We further devise a novel metric, called option gain, that is leveraged in the user interface and results in efficient and intuitive user interactions. Moreover, we take the initial steps toward providing descriptive answers that enable the users to assess the correctness of the answer to their question. We present the first question answering dataset that includes the verbalization of the answers. This resource empowers researchers to train and evaluate a variety of models to generate answer verbalizations. Our experiments exhibit satisfactory results by natural language generation models that are trained on our proposed dataset.

Acknowledgements

I would first like to thank my advisor, Prof. Jens Lehmann, without his support and guidance throughout my Ph.D. His door was always open whenever I ran into a trouble spot and welcomed me as I had many questions about my research or writing. His consistent encouragement empowered me with confidence through hardships. He stimulated me to undertake my research and steered me in the right direction. Furthermore, I applause his ability to maintain the Smart Data Analytics (SDA), a very diverse group of young researchers across genders and ethnicities with the highest moral and ethical standards.

I feel very fortunate to be a part of the SDA group. I am very grateful to all the members of the group at the University of Bonn and Fraunhofer IAIS. I enjoyed the support from my mentors Dr. Giulio Napolitano and Dr. Maria Maleshkova for their support and insightful discussions. I would like to express my gratitude to my colleagues for their helpful, profound and thoughtful discussions: Afshin Sadeghi, Debanjan Chaudhuri, Debayan Banerjee, Denis Lukovnikov, Endri Kacupaj, Fathoni A. Musyaffa, Firas Kassawat, Gaurav Maheshwari, Jason Armitage, Lars Reimann, Mehdi Ali, Mohnish Dubey, Mojtaba Nayyeri, Nilesh Chakraborty, Priyansh Trivedi. I would also like to extend my gratitude to Dr. Günter Kniesel, Dr. Diego Esteves, and Gezim Sejdiu with whom not only shared the office but also a lifetime friendship.

I would be remiss if I did not also put in writing my appreciation of the help provided by Dr. Giulio Napolitano, Jason Armitage, and Dr. Ricardo Usbeck for their valuable feedback, which greatly helped me to improve the quality of the thesis. Besides, I truly appreciate the effort of the rest of my thesis committee who accepted to be part of the examination process, in particular, Prof. Axel-Cyrille Ngonga Ngomo who kindly accepted to review this thesis.

I realize no amount of words can represent my appreciation for my lovely wife - Dr. Maryam Tavakol. The simplicity of my words fails to demonstrate the immensity of my gratitude for her constant support, encouragement, and sacrifices through stressful times as well as her insightful and inspiring scientific discussions.

This Ph.D. thesis is dedicated to my wonderful wife, Maryam Tavakol.

Contents

CHAPTER 1

Introduction

The emergence of Knowledge Graphs (KGs) as structured sources of information has engaged many researchers from various fields, including information retrieval and question answering. Openly available large-scale knowledge graphs such as DBpedia [1], Wikidata [2], YAGO [3] and EventKG [4], [5] have evolved as the key reference sources of information and knowledge regarding real-world entities, events and facts on the Web. The flexibility of RDF-based knowledge representation, the large-scale editor base of popular knowledge graphs and recent advances in automatic knowledge graph completion methods have led to a growth in the data and schema layers of these graphs at an unprecedented scale, with schemas including thousands of types and relations [6]. Hence, knowledge graphs now serve as a well-structured source of information in various applications such as information retrieval. However, the information contained in knowledge graphs is very hard to query, in particular due to their large scale and variety in schema descriptions - as well as the heterogeneity of the entities.

Consequently, Knowledge Graph based Question Answering (KGQA) systems are introduced as a key technology to facilitate end-users to query knowledge graphs using natural language interfaces. In recent years, a large number of KGQA approaches have been developed [7]. The objective of these approaches is to automatically comprehend a user question expressed in a natural language in order to obtain the answer from the underlying knowledge graph. In this work, we investigate the main-stream question answering approaches, and introduce various solutions to their pitfalls.

1.1 Motivation

ARPANET[1] is an early network in 1970s that implemented TCP/IP protocol, which later became the infrastructure for the internet. The preliminary idea of ARPANET was to connect geographically isolated networks. Soon after, even before the emergence of the World Wide Web (WWW), the first search engine, Archie [2], was introduced in 1990 to match file names with search terms entered by users, using regular expressions. Marked by the creation of the first website [3], the WWW unfolded in 1991 to meet the demand for sharing information among scientists and research institutions. A rapid rate of website creation led to the emergence of early web search engines such as W3Catalog [4] and

[1] Advanced Research Projects Agency Network

[2] Short for *Archives*

[3] http://info.cern.ch/

[4] https://www.w3catalog.com/

ALIWEB [5] in 1993. These primitive search engines offered simple textual search on their indices. Following the exponential growth of data on the web, the importance of search engines also increased as they acted as the entry portal for users to find information on the web. Users tend to view search engines as an intrinsic interface to find information using keywords or formulating a question posed in natural language. Nowadays, modern search engines are able to provide a concise answer(s) to simple questions such as *"what is the temperature in Bonn?"*. Nonetheless, given the unstructured nature of textual data available on the web, it has been a challenging task to comprehend more complex questions that need multi-step reasoning (for instance *"Which French presidents have studied in Paris?"*).

The mainstream approaches in the field of Knowledge Graph based Question Answering (KGQA) systems are semantic parsing based methods and end-to-end neural network based approaches. Semantic parsing approaches aim to transform a natural language question to an admissible formal query against the knowledge graph, whereas end-to-end systems directly return results without using a formal query language. Although recent advances in QA research have been beneficial for developing end-to-end systems [8], these approaches have several shortcomings. First, the obtained models from such end-to-end systems are not interpretable, hence conducting error analysis and improving the models based on intermediate outputs is very difficult.

Second, other approaches cannot re-use the internal - sometimes very sophisticated - components of these systems. This also restricts optimization using intelligent pipeline composition [9]. Third, the required amount of training data can be very high. Therefore, the focus of this thesis is to study Semantic-parsing based Question Answering (SQA) systems over knowledge graphs.

Semantic-parsing based Question Answering systems commonly segment the whole task into various subtasks usually performed sequentially, including shallow parsing, Named Entity Disambiguation (NED) and Relation Extraction (RE), and Query Generation (QG) among others [10]. Figure 1.1 depicts an exemplary question and shows that first, the natural question is passed to the shallow parser for detecting the entities and relations of the question. Once the words are labeled, the linker finds the corresponding entities/relations from the underlying knowledge graph. Finally, the top candidates are used to generate a formal query given the entity and relation mentions to retrieve the corresponding answer from the knowledge graph.

Despite the advancement in semantic-parsing approaches, it requires an immense effort to build a semantic-parsing based question answering systems, tailored for a specific domain. That is because most of the existing systems are build from scratch, even though they have a similar architecture. A few works offered the required infrastructure to build semantic question answering systems by assembling the internal components from the existing ones [9, 11]. However, they often perform poorly compared to highly coupled systems. That motivates us to work out a generic formal framework to define various tasks in semantic question answering, and design individual components for the critical tasks as a means to improve the overall accuracy of modular semantic question answering systems. Furthermore, we extend our formal framework by engaging users to interact with the system that leads to a remarkable increase in the performance of question answering systems, whilst keeping a high level of user satisfaction.

In the next section, we lay out a more detailed critical review of semantic question answering systems and summarise their shortcomings.

[5] http://www.aliweb.com/

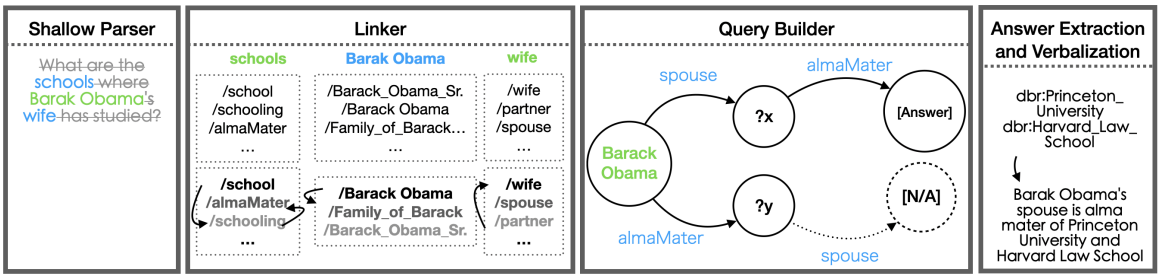


Figure 1.1: The overall pipeline of question answering system over knowledge graph, for the input question *"What are the schools where Barak Obama's wife has studied?"*

1.2 Challenges

We discuss four main categories of shortcomings in semantic questions answering:

During recent decades, researchers have had to devote ample effort to developing a question answering system mostly from scratch, even when the main objective is to enhance a specific task within the question answering system. As a result, an immense number of question answering systems have been introduced by the question answering community, with common characteristics and architectures. We believe this is one of the key challenges facing the question answering community and we argue that it is vital to carefully study each component/task, in order to find the **bottlenecks and enhance individual components** to support various aspects of complexity in questions that would lead to improving the overall accuracy of the QA pipeline.

This problem, however, is rooted in the fact that while we can segment semantic question answering into multiple components as presented in Figure 1.1. This rarely corresponds to true modularity in the architecture of implemented systems. Consequently, it results in the general inability by the wider community to successfully and efficiently build upon the efforts of past achievements. To tackle this problem, researchers introduced QA modular frameworks such as OKBQA [11] and Frankenstein [9] for reusable components. While this line of works provides various implementation, documentation and guidelines, it suffers from a lack of formalized definitions for semantic question answering systems. Hence, considering that Semantic Question Answering systems are evolved to follow a similar pipeline architecture consisting of individual components, we stress that the **formalization** of the architecture is necessary to study, analyze and re-use existing QA systems.

The third key shortcoming of QA modular frameworks is that they neglect the value of user interaction and its effect on increasing the performance of QA systems. Nonetheless, there are in fact KGQA systems such as IMPROVE-QA [12] and GQBE [13] that integrate user involvement through limited interaction with users, however, such systems are not flexible enough to allow for the replacement of their internal modules with the equivalent ones from the QA community. Considering the state of the art SQA systems, there is still remarkable room for improvement. This gap can be filled by a **modular question answering framework with an interactive schema** that is able to address the trade-off between user interaction and efficiency.

Last but not least, given that in general, the basic premise in question answering systems is that the user is seeking to find the answer to a question where the answer is not known, it is of the utmost importance to enable the user to verify the generated answer(s). Furthermore, considering that QA systems provide the answer(s) to a given natural language question with some level of uncertainty

about whether the true intention of the question is captured by the QA system, it is essential to enable the user to be able to **assert that the system has interpreted the question** in the from intended by the user without using any external source of information. Some researchers have used template-based approaches to generate a natural language equivalent to their candidate formal queries. However, current query verbalization methods have limited support for different features used by formal queries, hence, the quality of the result is degraded when formal representation does not fall into the pre-defined templates.

In the next section, we formulate a list of refutable research questions related to the challenges above in order to provide the ground for a set of measurable contributions that are introduced in Section 1.4.

1.3 Research Questions

Considering the aforementioned problem statement regarding the existing challenges in semantic questions, we compose the following research questions that are the basis of this thesis.

RQ-1 *How can semantic technologies assist in formalizing semantic question answering?*

This is the first step at the abstract level to provide the required foundation to define various components in SQA and their logical dependencies and data flow. It also enables a systematic way to support the integration of user feedback. Moreover, the formalization helps to better decouple various modules in QA systems. Consequently, it prompts re-usability and it enables the researcher to evaluate the performance of each step individually, which itself leads to identifying low performing modules. This takes us to the second research question:

RQ-2 *What are the low performing components in SQA and to what extent can the improvement of bottleneck components in SQA enhance the overall accuracy of the pipeline?*

We identify two narrowing components by studying recent advances in SQA. We review individual SQA systems and the responsible component for each task in order to elaborate on the shortcomings and weaknesses of the existing works. Furthermore, we present innovative approaches to mitigate the existing shortcomings as well as optimizing the performance of each component separately. Additionally, we evaluate the absolute performance improvement of the overall QA pipeline that arises from using our proposed approaches. Given an enhanced QA pipeline, we further investigate the users' involvement in the QA system as the next research question:

RQ-3 *Can user interaction with a QA pipeline improve the overall accuracy of the QA pipeline while maintaining user satisfaction?*

There is still remarkable room for improvement in QA systems and user interaction can be viewed as a compensation mechanism, in which the user can provide vital input to the system in order to truly capture the intention of the question, for instance by disambiguating a recognized entity mentioned in the question.

Hence, we extend formalization to embrace the interaction schema and present various strategies to balance the trade-off between user interaction and efficiency. Furthermore, we implement the interaction schema by re-using existing QA components and show that interactions with the user can be employed to increase the accuracy of QA systems. In addition, we conduct a user study to measure user satisfaction and success rate using a web-based interface.

Finally, motivated by the premise that users ask questions for which answers are not known to them, we compose the last research question as:

RQ-4 *Can answer verbalization enables users to verify the provided answer without using any external source of information?*

Considering that no question answering is able to always guarantee the answers it provides, it is vital to enable users to verify the provided answer. We review existing approaches and conduct a user study to discover their understandability from the users' perspective. Based on the user study, we extend a commonly used QA dataset to incorporate the verbalization of the answer and the formal query.

1.4 Contributions

The discussed research questions specify the main areas of focus in this thesis. In the following, we briefly present the high level contributions regarding each research question.

Contributions for RQ-1 *A formalization framework for pipelined semantic question answering system:* Various surveys [10, 14] studied challenges and tasks within the existing question answering systems. We exploit the similarities of the studied question answering systems to identify *de facto* standards tasks in the QA systems. We provide a formalization of SQA with a vision for enabling user interaction. We further devise an implementation of the formalization to support its applicability in constructing concrete SQA pipelines.

Contributions for RQ-2 *Identify bottlenecks and improve the neglected components in SQA:* We recognize two major points of failure in semantic question answering: Parsing and Query building. Although these are integral parts of any semantic question answering system, no one has provided a comprehensive study on their functionality or their effects on the overall performance of question answering pipelines. Parsing in question answering is one of the early tasks in the pipeline that elucidates the most informative utterances of the question. However, most QA systems use either rule-based approaches or existing out-of-the-box tools. Considering that the true labels for the parsing task are not commonly provided, we introduce a novel reinforcement learning approach to tackle the problem in a distantly supervised setting. We extensively evaluate our method and provide insights on its effect to enhance the performance of the QA pipeline.

In the next step, we focus on query building in semantic QA systems. Similar to the parsing task, it has been mostly realized in QA systems using a template-based method with limited generalizability. We develop a scalable algorithm that generates the candidate formal queries given a natural language question and sorts them with respect to their structural similarity to the input question.

Contributions for RQ-3 *Exert user interaction to improve QA performance:* We study recent works on involving users in the process of answering the questions. Other than final performance, one of the vital factors is the balance between the number of user interactions with the system versus user satisfaction. While users can help the system to find the expected intention of the question, it usually has a reverse correlation with user satisfaction. As a result, we adopt a cost-based decision tree to find a balance between user involvement and user satisfaction. Furthermore, we design various strategies

for the user interactions, in which users are faced with interaction options with different complexity levels. We perform an extensive evaluation in an oracle setting to discover the upper limit. We also conduct a user study to examine user satisfaction and performance in different strategies.

Contributions for RQ-4 *Answer verbalization in semantic question answering:* We conduct a user study based on existing approaches for answer verbalization to find out the comprehensibility of each one from the users' perspective as well as user experience. Our findings indicate that users are more in favor of natural language representation in comparison to other forms of representation. However, none of the existing approaches fully supports natural language representation of answers, mostly due to the lack of a proper question answering dataset with verbalization of the answer. Therefore, we annotate one of the widely used question answering datasets with the answer verbalization to provide the community with a valuable resource to develop various models for answer verbalization.

In this section, we summarise the principal contributions for each research question. In the upcoming section, we present the publications in which the contributions have been introduced.

1.5 Publications

The following list enumerates the publications in which the aforementioned contributions are realized. They form the basis of this thesis and are the source of various figures, tables and ideas that are presented in the rest of the thesis. It also gives a brief summary of the author's (Hamid Zafar) contributions to the papers.

1. **Hamid Zafar**, Giulio Napolitano, Jens Lehmann. *Formal query generation for question answering over knowledge bases.* In Proceedings of European Semantic Web Conference (ESWC), pp. 714-728. Springer, Cham, 2018.
   This paper originates from my idea that a simple or compound question can be mapped to walks within the knowledge graph. In addition to developing the idea, I conducted the experiments and carried out all the implementations.

2. **Hamid Zafar**, Giulio Napolitano, Jens Lehmann. *Deep Query Ranking for Question Answering over Knowledge Bases.* Zafar, H., Napolitano, G. and Lehmann, J., 2018, September. Deep Query Ranking for Question Answering over Knowledge Bases. In Proceedings of Joint European Conference on Machine Learning and Knowledge Discovery in Databases(ECML/PKDD) (pp. 635-638). Springer, Cham.;
   I proposed using a Tree-LSTM to exploit the structural similarity of the natural question and the candidate formal queries. I programmed the empirical studies for different experimental setups and evaluated various scenarios.

3. Abdelrahman Abdelkawi, **Hamid Zafar**, Maria Maleshkova, Jens Lehmann. *Complex Query Augmentation for Question Answering over Knowledge Graphs*. In Proceedings of OTM Confederated International Conferences" On the Move to Meaningful Internet Systems", International Conference on Ontologies, DataBases, and Applications of Semantics (ODBASE) 2019 (pp. 571-587). Springer, Cham.
   I supervised Abdelrahman Abdelkawi on his master thesis to extend our previous work on semantic query building to support various complex features such as ordinal and filter questions.

We developed the ideas together and the student performed the experiments. The base implementation was the SQG framework from my first publication.

4. **Hamid Zafar**, Maryam Tavakol, Jens Lehmann. *Distantly Supervised Question Parsing* In Proceedings of the 24th European Conference on Artificial Intelligence (ECAI) 2020;
I discovered that the performance of the linking component is highly dependent on the parsing step. However, due to the lack of true labels, supervised machine learning approaches fall short in this setting. Maryam Tavakol and I developed the idea together. I implemented the model and the experiments, while Maryam Tavakol worked on the problem setting and formalizing the approach.

5. Endri Kacupaj, **Hamid Zafar**, Jens Lehmann, Maria Maleshkova. *VQuAnDa: Verbalization QUestion ANswering DAtaset.* In Proceedings of European Semantic Web Conference (ESWC) 2020. This is joint work with Endri Kacupaj, a Ph.D. student at the University of Bonn. My contributions in this paper are as follows: the overview idea, problem definition, and reviewing the related work.

6. **Hamid Zafar**, Mohnish Dubey, Jens Lehmann, Elena Demidova. *IQA: Interactive Query Construction in Semantic Question Answering Systems.* Journal of Web Semantics Special Issue Language Technology and Knowledge Graphs 2020 (Under review). I proposed introducing a cost-sensitive decision tree to balance the usability and complexity of interaction options. I implemented the interactive schema, in addition to designing and evaluating the oracle experiments. Furthermore, I contributed to the formalized definition as well as conducting a user study.

In this section, we provided a summary of the contributions of the author within each publication. In the following, we present the structure of the thesis, which is adapted from the aforementioned publications.

1.6 Outline

We define the scope of the thesis, research questions, challenges as well as the contributions in Chapter 1. Chapter 2 summarizes the main theoretical and technical background, as well as a formalization for semantic question answering that is used throughout the thesis. In Chapter 3, we provide a historical overview of question answering systems within various fields and further zoom in on the most recent approaches that are of interest to this thesis.

In Chapter 5, we study shallow parsing, as the first task in semantic question answering. Due to the fact that the true labels are not available for shallow parsing on most of the existing question answering datasets, this task has been mostly neglected. However, in this chapter, we establish the importance of shallow parsing and its effect on the overall performance of question answering systems.

We focus on the task of query building and ranking in Chapter 6, motivated by the lack of comprehensive study on this particular task. We study various methods for query building tasks in semantic question answering and present their limitations. We further devise a scalable algorithm, called SQG, to generate candidate formal queries given a set of candidate link items. Our experiments reveal that the syntactical structure of the input question can be utilized to discover the most probable

formal counterpart query. We exert Tree-LSTM to exploit the structural similarity of the question and the candidate queries.

SQG is able to handle simple and compound questions as well as boolean and count forms. In Chapter 7, we focus on a wider range of questions that are currently less represented in the community of question answering such as questions that require ordinal and filter constraints. Hence, we propose an extension of SQG that allows further support for such features. Our proposed architecture can be easily adapted to support other questions, such as forms that require aggregation, if training data can be provided.

We provide a complete semantic question answering pipeline within an interactive framework in Chapter 8. In this pipeline, we seek user involvement in order to guide the system to capture the correct intention of the input question through the means of providing the user with various options that would restrict the search space. We measure user satisfaction within different strategies to balance the trade-off between user interaction and the overall performance of the pipeline in terms of accuracy.

Chapter 9 studies various approaches that enable users to verify the answer which is provided by the question answering system. We conduct a user study to evaluate the performance of the existing approaches as well as a novel full verbalized answer, in terms of accuracy and ease of use from the users' perspective. Given the promising outcomes from the verbalized approach, we introduce an innovative resource based on a well-known question answering datasets that empowers the community with train and test data.

Chapter 10 concludes our findings across the thesis and lays out the expansion points for the future.

CHAPTER 2

Preliminaries

In the previous chapter, we presented our motivations that were founded on existing challenges in the semantic parsing question answering community. In addition, we introduced four research questions along with a brief description of the related contributions.

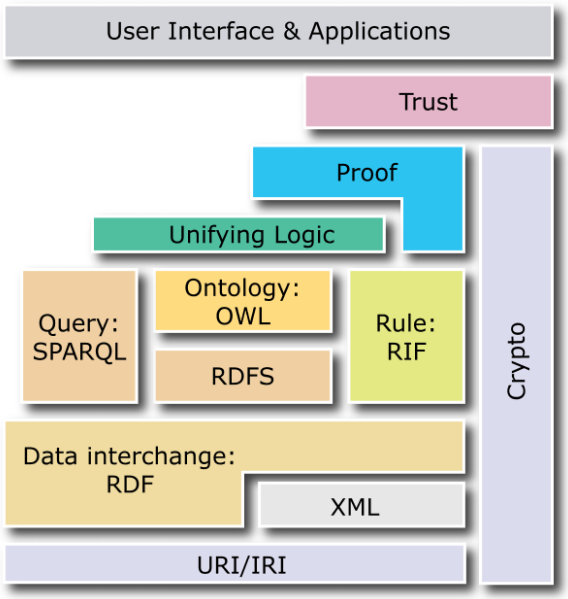
The purpose of this chapter is to present the basic foundations (formalization and technologies) for this thesis and describe the general concepts employed in our contributions. Section 2.1 covers the history of semantic web and the fundamental technologies that are used in semantic question answering such as RDF, OWL, knowledge graph and SPARQL. In Section 2.1.2, we further look into knowledge graphs as a well-structured source of information that is used in question answering over knowledge graphs. We review question answering systems that are based on semantic technologies and knowledge graphs, and their mainstream approaches in Section 2.2.

2.1 Semantic Web

The rapid growth of the Internet in the late 1990s connected millions of computers across the globe and provided a unique opportunity for its use as an infrastructure to share information. Motivated by the observation that many scientists have difficulties in accessing the information hosted on various computers, Tim Berners-Lee laid out a visionary proposal to facilitate information management [1]. Soon after, he implemented three fundamental technologies which are considered the foundation of the World Wide Web (WWW) as we know it today: (i) HyperText Markup Language (HTML), (ii) Uniform Resource Identifier (URI) and (iii) Hypertext Transfer Protocol (HTTP). While the web served its primitive objectives, Tim Berners-Lee made the assertion in 1991 that while documents on the web are comprehensible by humans, they are devoid of meaning to the computers. He floated the idea that a semantic layer could be added to documents if the information is encoded in a machine-readable format and linked across various documents. A decade later, the Semantic Web was introduced, using RDF (Resource Description Framework) and OWL (Web Ontology Language) technologies, to formally frame concepts to bring meaning and relations that can be understood by computers (for instance, knowledge extraction and reasoning).

In the following years, the World Wide Web Consortium (W3C) provided a set of standards that forms the building block of the semantic web stack (See Figure 2.1). In the bottom layer of the

[1] http://info.cern.ch/Proposal.html



[Source: http://www.w3.org/2007/03/layerCake.png]

Figure 2.1: The Semantic Web Stack

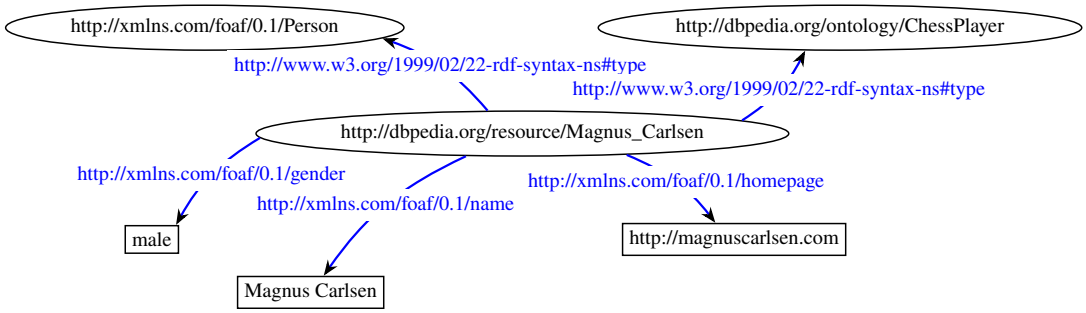


Figure 2.2: A Graph representation of Magnus Carlsen

Semantic Web stack, concepts are assigned a unique URI and XML (eXtensible Markup Language), and are used to create documents with semi-structured data. In the next layer, RDF enables the representation of simple statements (subject-predicate-object) based on URIs and XML. OWL is in the next rank: this is based on description logic and is used to promote reasoning and extend RDF with advance constructs such as constraints and characteristic properties. To enable query RDF-based data, there is SPARQL to retrieve the information stored in RDF knowledge bases.

The Semantic Web Stack and its underlying building blocks provide the infrastructure referenced throughout the thesis. We elaborate on them in detail in the following sections.

2.1.1 RDF and OWL

W3C introduced the Resource Description Framework (RDF) as a metadata data model in 1999. It enables expressing statements in the form of subject-predicate-object (a.k.a. triples) about resources, where each triple conveys a relationship, designated by the predicate, between the object and subject.

Figure 2.2 depicts a graph that relates to a resource about a person called “Magnus Carlsen" who is a male chess player. In each triple, an object is always represented by a URI, while the subject can be either a URL or a literal. Note that a collection of RDF statements forms a labeled directed multi-graph and can be stored in various formats such as RDF Turtle [2] (See Example 1) or RDF/XML syntax (See Example 2).

Example 1 (RDF Turtle Syntax)

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .

<http://dbpedia.org/resource/Magnus_Carlsen> a foaf:Person,
      <http://dbpedia.org/ontology/ChessPlayer> ;
    foaf:gender "male" ;
    foaf:homepage <http://magnuscarlsen.com> ;
    foaf:name "Magnus Carlsen" .
```

Example 2 (RDF/XML Syntax)

```
<?xml version="1.0" ?>
<rdf:RDF
   xmlns:foaf="http://xmlns.com/foaf/0.1/"
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
  <rdf:Description rdf:about="http://dbpedia.org/resource/Magnus_Carlsen">
    <rdf:type rdf:resource="http://xmlns.com/foaf/0.1/Person"/>
    <rdf:type rdf:resource="http://dbpedia.org/ontology/ChessPlayer"/>
    <foaf:name>Magnus Carlsen</foaf:name>
    <foaf:gender>male</foaf:gender>
    <foaf:homepage rdf:resource="http://magnuscarlsen.com"/>
  </rdf:Description>
</rdf:RDF>
```

Though simple, the powerful RDF data model is able to store knowledge about disparate concepts and their relationships in a way that can be processed by computers as well as humans. Furthermore, in order to structure RDF resources, vocabularies are used to define concepts, classes and properties. Consequently, *RDF vocabulary* is introduced to bring a common structure to RDF resources. RDF vocabulary defines a simple language that specifies a collection of basic classes and properties. RDF Schema (RDFS) extends the RDF vocabulary and further defines object-oriented characterizations (such as “Type", “Class" and “Instance") to facilitate the definition of RDF vocabularies.

However, more complex vocabularies (a.k.a. ontologies) can be supported by Web Ontology Language (OWL) which is based on description logics that extends RDF vocabulary and RDFS. It allows consistency verification as well as applying reasoning techniques. W3C introduced OWL in three different flavors: OWL Lite, OWL DL and OWL Full to support various levels of complexity.

Regardless of the vocabulary, RDF-based statements are stored in RDF triple-stores (for instance Virtuoso and Stardog), which can be queried using a formal query language. In the next section, we review the application of RDF/OWL for knowledge representation.

[2] http://www.w3.org/TeamSubmission/turtle/

2.1.2 Knowledge Graph

A Knowledge Base (KB) is a knowledge storage system that facilitates preserving and retrieving structured and semi-structured data. In contrast to relational data models where semantic annotation of information cannot be easily done, KBs use ontologies (e.g. RDF/OWL) that enable enriching the data model and consequently reasoning over the information [15]. Triple-based KBs describe knowledge using interlinked descriptions about the entities such that it forms a semi-connected graph. Hence they are also referred to as Knowledge Graphs (KGs) in the literature.

Due to the well structure of KGs that permits querying and reasoning, there has been an emergence of large-scale KGs over the last few years. This resulted in many domain-specific knowledge graphs in various areas such as Knowlife [16] in health and life sciences, as well as a few very large scaled open-domain KGs. Open-domain KGs capture common knowledge about various entities. Freebase [17], DBpedia [1] and Wikidata [18] are amongst the most prominent ones that are publicly available. DBpedia is created using various information extraction techniques to convert the content of Wikipedia, the largest online encyclopedia, to RDF triples. In contrast to the full-text search capability of Wikipedia, DBpedia can be used to query complex questions that require inference and reasoning.

Formally, a *knowledge graph* $\mathcal{KG} = (V, L, E, T)$ consists of a set V of entities, a set L of literals, a set E of properties and a set $T \subseteq V \times E \times (V \cup L)$ of triples.

The entities in V represent real-world entities and concepts. The properties in E represent relations connecting two entities or an entity and a literal value.

Regardless of the ontology, RDF-based KGs are stored in RDF triplestores (for instance Virtuoso and Stardog), which can be queried using a formal query language. In the next section, we review SPARQL, which is the official recommendation of W3C.

2.1.3 SPARQL

SPARQL (SPARQL Protocol and RDF Query Language) is a querying language for RDF-based datastores and holds similarities with the role of SQL as it pertains to relational databases. The most essential idea of SPARQL is to express the desired query as a collection of triple-pattern statements, where each triple-pattern resembles an RDF-triple where any part of the triple can be a variable. This pattern collection should for a match with a subgraph from the knowledge graph.

There are four main types of SPARQL query: (i)*SELECT* query that retrieves the specified variables from the query match pattern, (ii) *ASK* query that returns True or False depends on whether the query pattern exists in the knowledge graph, (iii) *CONSTRUCT* command that creates RDF statements where the variables are substituted by their values from the matched patterns, (iv) *DESCRIBE* query that provides RDF statements to describe the matching resources.

A *SELECT* query is shown in Example 3 that retrieves the number of male chess players who have a homepage.

Example 3 (SPARQL Query)

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>

SELECT COUNT(?person) where {
    ?person rdf:type <http://dbpedia.org/ontology/ChessPlayer>.
    ?person <http://xmlns.com/foaf/0.1/gender> "male"@en.
```

```
    ?person foaf:homepage ?website.
}
```

The triple patterns in SPARQL query can be augmented with optional patterns (`OPTIONAL`), pattern alternatives (`UNION`), result restrictions (`FILTER`), and aggregation functions. For instance, by adding

```
FILTER regex(str(?website), ".com")
```

to the Example 3, it would retrieve the number of male chess players that have a website with ".com" extension.

SPARQL is an extremely expressive and powerful querying language but end-users are not equipped with the necessary skills to directly express their desired question as a formal query. Consequently, question answering systems are introduced that provide a user-friendly interface for the user to ask their questions in natural language with no need to understand the underlying technologies. In the next section, we briefly review mainstream knowledge graph based question answering approaches.

2.2 Question Answering using Knowledge Graphs

Openly available large-scale knowledge graphs such as DBpedia [19], Wikidata [2], YAGO [3] and EventKG [4] [5] have evolved as the key reference sources of information and knowledge regarding real-world entities, events and facts on the Web. The flexibility of RDF-based knowledge representation, the large-scale editor base of popular knowledge graphs and recent advances in the automatic knowledge graph completion methods have led to a growth in the data and schema layers of these graphs at an unprecedented scale, with schemas including thousands of types and relations [6]. As a result, the information contained in knowledge graphs is hard to query, in particular due to the large scale, the heterogeneity of the entities and the variety of their schema descriptions.

Question Answering (QA) is the key technology for enabling end-users to query knowledge graphs using natural language interfaces. The well-structured information in knowledge graphs allows for developing QA models that are able to query or reason over such knowledge graphs to support complex questions (such as multi hop questions, aggregation and ordinal questions [20]) going beyond the types of questions that can usually be successfully answered using unstructured text corpora. In recent years, a large number of QA approaches have been developed [7]. The objective of these approaches is to automatically interpret a user question expressed in a natural language as a semantic query (typically expressed in the SPARQL query language), which is then executed against the knowledge graph to obtain the results.

The mainstream approaches in the field of Knowledge Graph based Question Answering (KGQA) systems are semantic parsing based methods and end-to-end neural network based approaches. Semantic parsing approaches aim to transform a natural language question to an admissible query against the knowledge graph, whereas end-to-end systems directly return results without using a formal query language. Although recent advances in QA research have been beneficial for developing end-to-end systems [8], these approaches have several shortcomings. First, the obtained models from such end-to-end systems are not interpretable, which also means that conducting error analysis and improving the models based on intermediate outputs is very difficult. Second, other approaches cannot re-use the internal - sometimes very sophisticated - components of these systems. This also restricts optimization using intelligent pipeline composition [9]. Third, the required amount of training data can be very high.

Therefore, we aim to study modular question answering systems over knowledge graphs (KGQA). As the complexity of such systems may be considerable, it is common practice to segment the whole task into various subtasks usually performed sequentially, including Named Entity Disambiguation (NED), Relation Extraction (RE) and Query Generation (QG) among others [10]. In the next section, we formalize the concept of semantic Question Answering pipelines.

2.3 Summary

In this chapter, we introduced the fundamental concepts and technologies referenced in this thesis. We further discussed the main categories of question answering systems using knowledge graphs. Particularly, we elaborate semantic question answering as it is the focus of the thesis.

In the next chapter, we present state of the art question answering approaches and highlight the major bottlenecks found in them. This provides the motivation for later chapters.

CHAPTER 3

Question Answering Overview

We reviewed the common concepts and technologies that are used throughout the thesis in the preceding chapter. In this chapter, we focus on question answering systems as a whole. Additional to this, we also study the most relevant related works and state of the art methods within each chapter.

Question answering systems have become popular in recent decades and have received remarkable attention from various communities such as those found in the areas of information retrieval, natural language processing and semantic web.

First, we briefly discuss the advancement of question answering systems that are emerged from the natural language processing perspective. Then, in Section 3.2, we focus on the question answering systems that emerged from the information retrieval community that mostly focused on simple string similarity approaches. In Section 3.3, we study important related works and review the major advancements and shortcomings of state-of-the-art question answering approaches using knowledge graphs and the semantic web. Finally, Section 3.4 explores existing commonly used semantic question answering datasets and their characteristics.

3.1 Question Answering Over Text

In late 1990s, open domain questions answering systems were emerging that was initiated by TREC [1] campaign in 1999. At the first event, 200 questions from a set of documents were used for the evaluation. Participant systems were allowed to provide an answer limited to 250 characters. In subsequent events, both the number of questions and the size of corpora were increased, while shorter answers were considered preferable [21]. Furthermore, the complexity of questions was also expanded to include various types of question such as list, factoid, temporal and reasoning.

Furthermore, the growth of the web in the following year brought an unprecedented amount of unstructured data. Consequently, there was a new trend in the question answering community to leverage the web as the main source of information. The content from the web can be viewed as similar to the textual documents that are used on TREC campaigns. However, the main difference is that the scale of the input documents acts as the source of knowledge. This is also reflected in the common architecture of document-based question answering system in contrast to web-based question answering methods. Both have a similar general architecture, which consists of three consequential

[1] https://trec.nist.gov/

steps: (1) transformation of natural language questions to formal queries, (2) searching the source of information, and (3) extraction of the answer from selected documents. While document-based question answering systems benefit from information retrieval methods to look into the source corpus, web-based question answering systems usually consult search engines to find the relevant documents.

Since this topic is not the main focus of the thesis, we refrain from elaborate on this point. However, we refer the interested reader to [21–23] for surveys on question answering system based on textual data.

In the following section, we study recent trends in the question answering systems based on structured information, as opposed to the textual corpus.

3.2 Question Answering Over Databases

The preliminary development of question answering systems over databases dates back to the 1960s. An early study [24] reviews the merits and deficiencies of natural language interfaces in relation to databases in comparison to other interfaces such as formal query languages and graphical interfaces. Most of such systems [25, 26] are built for a particular domain and based on a specific database with no consideration for further modification or adoption. The main aim of these approaches is to translate the input question that is posed in natural language into a formal query that is compatible with the underlying database. Although the main motivation of such approaches is that the users need not learn either the formal query language or the data model of the underlying database, they mainly exert pattern matching based on string similarity metrics. As a result, these approaches could not be easily leveraged in other applications.

In the following works, researchers work on separating the language understanding module from the domain-dependent answer retrieval module. In this line of works [27–29], the input question is transformed into a logical language, and it is further used by domain-dependent modules to evaluate the logical expressions and map it into a query language to retrieve the final answer from the target database. Most recently, researchers introduced end-to-end neural network based models to transform the natural language directly to SQL statements [30–33].

We refer the interested readers to [24, 34, 35] for surveys on natural language interfaces to databases.

3.3 Question Answering Over Knowledge Graphs

While there are a vast amount of unstructured data available on the web, answering questions that require reasoning or fetching information from multiple documents is not currently well supported by state of the art approaches. Consequently, a new class of question answering was introduced that was based on semantic enabled structured datastores. The main idea is to build a factual knowledge graph that acts as the source of information for the downstream applications such as question answering. The early instances of such knowledge graphs (eg Omnibase [36], Powerset, and True Knowledge) were mostly proprietary and focused on a specific domain.

However, the steady growth of semantic web technologies started a movement in the field of question answering that resulted in hundreds of academic and industrial works at large scale. Ontology based datastores are knowledge bases that are shipped with a semantic-enabled expressive ontology that facilitate answering complex questions (e.g. questions that require reasoning). Researchers study the behavior of end-users in terms of usability when the users were provided with various interfaces to

query knowledge bases such as keyword, menu-guided, graphical representation and full sentence [37]. The results show that casual users prefer the full sentence interface over other types of interface.

Depending on the scope of the underlying source of information, we can categorize question answering systems into open domain and domain specific. Domain-specific systems tend to be highly supervised [38–41]. For instance, Ferrandez [41] introduced QACID, a question answering system in the domain of cinema. They collected user queries and grouped them into various clusters, where each cluster contains various natural representations of a formal query, which are created manually as well. Note that such systems resemble the NLI systems that are discussed in Section 3.2. However, the main feature that distinguishes them from NLI on databases is the advantage of using the ontology data model as opposed to the relational data model. Ontologies are far more flexible and expressive than legacy relational data models and they facilitate the transformation of natural language questions into formal queries. The main drawback of such question answering approaches is that they are limited to a specific domain, and further application of it requires extensive manual work such as defining domain-specific grammar [38, 39], extracting patterns lexicons [41], domain-dependent guided interface and user interfaces [42], domain-dependent dictionaries [43] (See [34, 44, 45] for more details).

There are two reasons that have led to a growing area of research that exploits the advantages of such large scale open-domain knowledge graphs for various tasks such as question answering: First, the limited focus and applicability of domain-specific question answering systems; and second, the emergence of large scale open-domain knowledge bases such as DBpedia [1], Freebase [17] and Wikidata [2]. The *Question Answering over Linked Data* (QALD) workshop [2] in 2011 was the first attempt to organize a campaign to evaluate question answering systems that are based on open-domain knowledge graphs. The participant QA systems in the first and second QALD events were studied in a survey [46] that summarized various techniques and approaches used in question answering systems. Freitas et al. [47] reviewed information retrieval based methods as well as natural language based approaches to query linked datastores, and analyzed them based on the core challenges that they defined: query expressiveness, usability, Vocabulary gap, entity linking, semantic tractability. Hoffner et al. [14] extended the survey by Lopez et al. [46] and identified 62 KGQA systems that were published between 2010 and 2015. Similar to the challenges discovered in [47], Hoffner et al. [14] identified seven main challenges in KGQA, namely: lexical gaps, ambiguities, multilingualism, complex queries, distributed knowledge, procedural, temporal and spatial questions, and templates. They further analyze the KGQA system with respect to the aforementioned challenges and provides insights about future directions such as general QA frameworks to re-use existing approaches. Most recently, Diefenbach et al. [10] revisited the most advanced techniques and approaches that are employed in KGQA systems. They identified five consecutive tasks: question analysis, phrase mapping, disambiguation, query construction and querying knowledge bases. They further discuss how various KGQA systems carry out their tasks.

3.4 Knowledge Graph based Question Answering Datasets

Table 3.1 summarizes the features of all existing QA datasets over KGs. The early Knowledge Graph based Question Answering (KGQA) systems were mostly template or rule-based systems with limited learnable modules [48, 49], mainly due to the fact that the existing QA datasets were small-scaled [50].

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

Consequently, researchers in the QA community are working on expanding QA datasets from two perspectives: (i) size - to support machine learning approaches that need more training data [51] and (ii) complexity - to move on from simple factoid questions to complex questions (e.g. multi-hop, ordinal, aggregation, etc) [52]. Note that while there are some QA datasets that are automatically generated [53], most QA datasets are manually created either by (i) using in-house workers [54] or crowd-sourcing [55] or (ii) extracting questions from online question answering platforms such as search engines and online forums [52]. The goal is to create datasets that are representative in terms of the types of questions that users are likely to ask.

These large-scale and complex QA datasets enable researchers to develop end-to-end learning approaches [56] and support questions with various features of varying complexity [20]. As a result, the main focus of many competitive QA methods is to enhance the performance of QA systems in terms of the accuracy of answer(s) retrieval.

Table 3.1: Summary of QA datasets over knowledge graphs

Dataset	KG	Size	Year	Formal Rep.	Creation
Free917 [50]	Freebase	917	2013	SPARQL	Manual
WebQuestions [52]	Freebase	5810	2013	None	Manual
SimpleQuestions [51]	Freebase	100K	2015	SPARQL	Manual
WebQuestionsSP [57]	Freebase	5810	2016	SPARQL	Manual
ComplexQuestions [58]	Freebase	2100	2016	None	Manual
GraphQuestions [59]	Freebase	5166	2016	SPARQL	Manual
30M Factoid Questions [53]	Freebase	30M	2016	SPARQL	Automatic
QALD (1-9) [3]	DBpedia	500	2011-2018	SPARQL	Manual
LC-QuAD 1.0 [54]	DBpedia	5000	2017	SPARQL	Manual
ComplexWebQuestions [60]	Freebase	33k	2018	SPARQL	Manual
ComQA [61]	Wikipedia	11k	2018	None	Manual
SimpleDBpediaQA [62]	DBpedia	43K	2018	Inferential Chain	Manual
CSQA [63]	Wikidata	200K	2018	Entities/Relations	Manual
LC-QuAD 2.0 [55]	Wikidata	30K	2019	SPARQL	Manual
FreebaseQA [64]	Freebase	28K	2019	Inferential Chain	Manual

3.5 Summary

In this chapter, we reviewed the development of question answering systems from information retrieval, natural language processing and semantic web communities. In addition, we highlighted the shortcomings of each perspective. Last but not least, we took a closer look at semantic question answering datasets and highlighted the trend towards more complex and large-scale datasets.

In the forthcoming chapters, we zoom in on individual research contributions. In the following chapter, we focus on **RQ-2**, and present an innovative model for shallow parsing tasks in question answering systems and provide experimental evidence that highlights the effect of the task on downstream tasks such as entity and relation disambiguation.

CHAPTER 4

Formalization of a Semantic Question Answering Pipeline

This chapter provides a formalization of a semantic question answering pipeline at an abstract level that will be referred to throughout the rest of this thesis. First, we present the required specifications for formalization and discuss the existing works. Furthermore, we present our formalization of a QA pipeline along with a probabilistic foundation to estimate the likelihood of question interpretations. A realization of the QA pipeline is presented in Chapter 8, where it is extended with a user interaction scheme that incorporates user feedback.

4.1 Requirements for Formalization

A semantic Question Answering pipeline (denoted as "QA pipeline" in the following) transforms a user question specified in a natural language in a semantic query for the target knowledge graph. The idea of a generic framework to promote usability for semantic question answering pipelines is not new. OpenQA [65] is an extensible open-source framework that facilitates the implementation and evaluation of QA pipelines, however, they must be implemented in Java. QALL-ME [68] framework is context-aware (space and time of question) and it has a service-oriented architecture that makes it language agnostic. OKBQA framework [11] is a collaborative platform to instantiate semantic question answering using existing components. Similar to QALL-ME, it also has a service-oriented architecture that allows users to choose their preferred programming language. Most recently, Qanary QA vocabulary [66] and QA estro [67] are proposed to further advance not only reusability but also the interoperability of question answering components using semantic technologies such as vocabularies and ontologies.

However, we require an abstract framework to describe a generic QA pipeline that is not limited to any particular technology or programming language. Furthermore, we expect it to enable user interaction in order to guide the QA system toward capturing the correct intention of the input question.

While the existing QA frameworks facilitate reusability and interoperability, they fall short in integrating user's feedback. Thus, we present a formalization of a QA pipeline that abstracts from the particular implementation, with a vision to enabling user interaction to guide the QA system effectively.

4.2 Basic Concepts

The goal of Semantic Question Answering is to transform a user question expressed in a natural language into a semantic query for the target knowledge graph.

We formally defined the concept of a knowledge graph in Section 2.1.2. In the following, we formalize the concepts of *user questions* and *semantic queries*.

A *user question* $Q = (q_{NL}, QN)$ is a tuple that represents user input. q_{NL} is the initial user question expressed in a natural language. $QN = \{n_1, \ldots, n_m\}$ is a multiset of information nuggets mentioned in the user question.

Information nuggets can include surface forms of named entities, concepts, and relations mentioned in q_{NL}. Information nuggets can be extracted from q_{NL} using information extraction techniques such as shallow parsing.

For example, consider the question:

$$q_{NL} = \textit{"List software that is written in C++ and runs on Mac OS."}$$

This question can be transformed into the following set of information nuggets:

$$QN = \{\text{"}software\text{"}, \text{"}written\text{"}, \text{"}C++\text{"}, \text{"}runs\text{"}, \text{"}Mac\ OS\text{"}\}.$$

In the process of Semantic Question Answering, information nuggets mentioned in the user question are interpreted as elements of the knowledge graph. A *nugget interpretation ni* is a mapping from an information nugget $n \in QN$ to an element of the knowledge graph $\mathcal{KG}$. An information nugget can be interpreted as an entity, a literal, a property, a single triple, or a set of triples.

For example, the nugget interpretation:

$$ni_0 = \{\text{"}software\text{"} \mapsto dbo : Software\}$$

maps the information nugget *"software"* to the entity
"dbo:Software" of the knowledge graph. Other examples of nugget interpretations include:

$$ni_1 = \{\text{"}written\text{"} \mapsto dbo : programmingLanguage\},$$
$$ni_2 = \{\text{"}C++\text{"} \mapsto dbr : C{+}{+}\},$$
$$ni_3 = \{\text{"}runs\text{"} \mapsto dbo : operatingSystem\},$$
$$ni_4 = \{\text{"}Mac\ OS\text{"} \mapsto dbr : Mac\_OS\}.$$

When an SQA pipeline transforms the user question Q into a semantic query, the pipeline components can generate intermediate interpretation results that include several nugget interpretations. We refer to such intermediate results as *partial question interpretations*. More formally:

A *partial question interpretation* $QI = \{ni_1, \ldots, ni_r\}$ is a set of nugget interpretations that interpret a (sub)set of the information nuggets contained in QN.

For example, a partial question interpretation

$$
\begin{aligned}
QI = \{ & \\
& \{\text{"}C{+}{+}\text{"} \mapsto dbr : C{+}{+}\}, \\
& \{\text{"}Mac\ OS\text{"} \mapsto dbr : Mac\_OS\}\}
\end{aligned}
$$

includes specific interpretations of two information nuggets representing entity surface forms in the user question.

Partial question interpretations serve as a basis for building semantic queries.

A semantic query CQI is a *complete question interpretation* that represents the user question as a whole. Intuitively, a CQI includes the elements of the knowledge graph that correspond to the nugget interpretations and connects them in a graph pattern.

Formally, a *complete question interpretation* $CQI = (QI, AT, QG)$ is a tuple that consists of a set of nugget interpretations QI, an answer type AT and a query graph QG. The answer type AT is an element of {"ASK", "SELECT", "COUNT"}. Given a knowledge graph $\mathcal{KG} = (V, L, E, T)$, a query graph $QG = (V', L', E', U, T')$ is a graph pattern such that: $V' \subset V$ is a set of entities, $L' \subset L$ is a set of literals, $E' \subset E$ is a set of properties, U is a set of variables and $T' \subset (V' \cup U) \times (E' \cup U) \times (V' \cup L' \cup U)$ is a set of triple patterns.

For example, $CQI_1 = (QI_1, QG_1, AT_1)$, is a complete question interpretation of the example question presented above, where: $AT_1 = \text{"}SELECT\text{"}$,

$$
\begin{aligned}
QI_1 = \{ & \\
& \{\text{"}software\text{"} \mapsto dbo : Software\}, \\
& \{\text{"}written\text{"} \mapsto dbo : programmingLanguage\}, \\
& \{\text{"}C++\text{"} \mapsto dbr : C{+}{+}\}, \\
& \{\text{"}runs\text{"} \mapsto dbo : operatingSystem\}, \\
& \{\text{"}Mac\ OS\text{"} \mapsto dbr : Mac\_OS\}\},
\end{aligned}
$$

and

$$
\begin{aligned}
& QG_1 = (\\
& V' = \{dbo : Software, dbr : C{+}{+}, dbr : Mac\_OS\}, \\
& L' = \emptyset, \\
& E' = \{rdf : type, dbo : programmingLanguage, \\
& dbo : operatingSystem\}, \\
& U = \{?uri\}, \\
& T' = \{ \\
& \quad ?uri\ rdf : type\ dbo : Software, \\
& \quad ?uri\ dbo : programmingLanguage\ dbr : C{+}{+}, \\
& \quad ?uri\ dbo : operatingSystem\ dbr : Mac\_OS.\}).
\end{aligned}
$$

To retrieve answers from a knowledge graph, a complete question interpretation can be translated into a query in the SPARQL query language[1]. For example, the following SPARQL query corresponds to the complete question interpretation of the example question presented above:

$$
\begin{aligned}
&SELECT\ ?uri\ WHERE\ \{\\
&?uri\ rdf:type\ dbo:Software.\\
&?uri\ dbo:programmingLanguage\ dbr:C{+}{+}.\\
&?uri\ dbo:operatingSystem\ dbr:Mac\_OS.\}
\end{aligned}
$$

Note that a complete question interpretation does not necessarily include interpretations of all information nuggets extracted from the user question. This is because information nuggets in QN can potentially contain redundant information.

4.3 Semantic Question Answering Pipeline

A typical Semantic Question Answering pipeline consists of: 1.) a shallow parser plc_{sp} constructing information nuggets, 2.) linkers plc_{link}: here we support different options of entity, relation and class linking separately or jointly – so there can be one or multiple linkers, and 3.) a query builder plc_{qb} creating complete question interpretations.

More formally, a *Semantic Question Answering pipeline* PL is a list of components, where each component $plc \in PL$ implements an *interpretation function*. The aim of an interpretation function is to incrementally transform the user question into candidate question interpretations.

$$PL = \Big(\big(plc_{sp}\big), \big((plc_{link_1}), \ldots, (plc_{link_p})\big), \big(plc_{qb}\big)\Big). \tag{4.1}$$

A pipeline component plc can generate multiple candidate interpretations.

The component plc_{sp} is a specific shallow parsing component at the first step of the pipeline, which transforms the user question into a set of information nuggets: $plc_{sp} : \mathbb{Q}_{\mathbb{NL}} \mapsto \mathbb{QN}$, where $\mathbb{Q}_{\mathbb{NL}}$ is the set of natural language questions, and $\mathbb{QN}$ is the set of information nuggets.

A plc_{link} component takes the user question and, optionally, an interpretation produced by the previous pipeline component as an input and produces a set of partial interpretations as an output: $plc_{link} : \mathbb{Q} \times \mathbb{QI} \mapsto \mathcal{P}(\mathbb{QI})$, where $\mathbb{Q}$ is the set of questions, $\mathbb{QI}$ is the set of partial question interpretations, and $\mathcal{P}$ is the power set constructor. Examples of interpretation functions of the components plc_{link} include entity linking, relation linking, and class linking. There can be a single joint linking step or multiple individual linking steps. By supporting all of those scenarios, the interaction framework described in this chapter can be applied to a broader range of existing SQA frameworks.

The component plc_{qb} is a specific query building component at the last step of the pipeline, which transforms a partial question interpretation QI into one or more complete question interpretations, i.e. $plc_{qb} : \mathbb{QI} \mapsto \mathbb{CQI}$, where $\mathbb{CQI}$ is the set of complete question interpretations. Each question

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

interpretation $QI \in \mathbb{QI}$ and $CQI \in \mathbb{CQI}$ is associated with a confidence score generated by the corresponding pipeline component.

Conceptually, as an SQA pipeline processes the user question, it incrementally generates a hierarchy of question interpretations, where partial question interpretations are the intermediate nodes, and complete question interpretations are the leaf nodes.

4.4 Probability of Complete Question Interpretations

To estimate the probability $P(CQI|Q, \mathcal{KG})$ of the complete question interpretation $CQI = (QI, AT, OG)$ to be intended by the user, given the user question $Q = (q_{NL}, QN)$ and the knowledge graph $\mathcal{KG}$, we consider the following factors: 1) the likelihood of the partial question interpretation $QI = \{ni_1, \ldots, ni_r\}$ from which CQI was composed by the SQA pipeline, represented as $P(QI|Q, \mathcal{KG})$ and 2) the probability of the graph structure QG of the semantic query given the linguistic structure of the user question q_{NL}, represented as $P(QG|q_{NL}, \mathcal{KG})$.

For mathematical simplification, similar to Naïve Bayes, we assume that the probabilities of the nugget interpretations in the QI from which CQI is constructed, as well as the structure QG of the resulting semantic query are mutually independent. Although the resulting probability estimation is potentially not very precise, it leads to an adequate prediction of query relevance, as shown by our experiments.

Then the probability $P(CQI|Q, \mathcal{KG})$ of the complete question interpretation CQI can be estimated as:

$$P(CQI|Q, \mathcal{KG}) \propto \left(\prod_{ni_i \in QI} P(ni_i|Q, \mathcal{KG}) \right) \times P(QG|q_{NL}, \mathcal{KG}).$$

We estimate $P(ni_i|Q, \mathcal{KG})$ using the confidence score provided by the pipeline component that generates the nugget interpretation ni_i. $P(QG|q_{NL}, \mathcal{KG})$ is also estimated by the related pipeline component to capture the similarity of CQI and q_{NL}; for instance, using the structural similarity between the graph structure of CQI and the parse tree structure of the user question q_{NL}.

4.5 Summary

In this chapter, we provided an overview of the existing approaches to generalize and reuse QA pipelines. We pointed out that user interaction is not integrated into any of those methods. Hence, we presented a formalization of semantic question answering over the knowledge graph at an abstract level to address **RQ-1**. This formalization enables us to integrate user feedback efficiently. It also allows us to break down the task into manageable pieces.

In the next three chapters, we present various approaches to enhance individual components in semantic question answering based on the presented formalization. In Chapter 8, we instantiate a fully functioning QA pipeline using the formalization presented in this chapter. We further devise an interaction schema to utilize user feedback to guide the QA pipeline toward correct interpretation in a user-friendly manner.

CHAPTER 5

Shallow Parsing

We presented the research problems as well as the research questions in Chapter 1. In Chapter 2, we laid out the foundations of the thesis as well as the overall formalization to address the first research question. We reviewed the main-stream approaches in semantic parsing question answering using knowledge graphs in the previous chapter.

In this chapter, we concentrate on the **RQ-2** that is to identify and enhance bottleneck components in semantic question answering. Specifically, we focus on the (shallow) parsing task, which is usually the essential first component of a typical KGQA pipeline (cmp. Figure 5.2). In the following, we first introduce the shallow parsing task on a running example shown in Figure 5.1 and call attention to its challenges. The Section 5.2 presents the related work on entity/relation detection and linking approaches, as well as related techniques used in QA systems. We present the details of our proposed approach in Section 5.3 followed by the empirical study in Section 5.4, and Section 5.5 concludes our contributions toward the second research question.

5.1 Introduction

Although shallow parsing is a key step in the process, and almost all recent approaches for complex QA tasks over KGs use shallow parsing, it has received little attention in previous works. The problem is addressed either by a simple technique for pattern/template matching [48, 69] or an existing shallow parser from the Natural Language Processing (NLP) community [70]. However, the well-known shallow parsing methods from NLP such as [71, 72] are usually not suitable for QA systems. A core reason for this is the difference in the syntactical structure of questions compared to normal sentences. Most text corpora contain a low percentage of questions and, naturally, the models trained on those are not properly tuned for questions. Additionally, these methods are designed to provide a fine granularity level of parsing in the sentences, while we require a more shallow decomposition. This motivates designing a proper shallow parser specific for QA systems.

Figure 5.1 (top) depicts an exemplary (question, formal-query) pair followed by the desired annotations (bottom) that provide the most relevant information required for answer retrieval. The output of parsing is further used in the next step of the overall procedure of question answering over KGs which is displayed in Figure 5.2. The figure shows that first, a natural question is passed to shallow parser for detecting the entities and relations of the question. Once the words are labeled, the linker finds the corresponding entities/relations from the underlying knowledge graph. Finally, the top

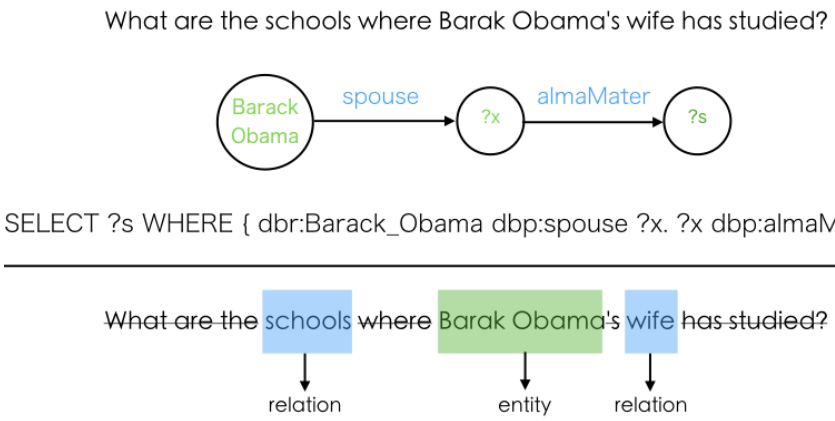


Figure 5.1: An example of shallow parsing task for a given question (with a small typo) and its corresponding formal query in a knowledge graph.

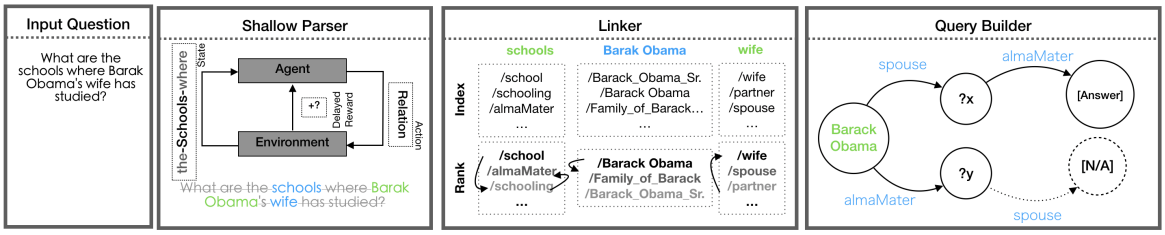


Figure 5.2: The overall pipeline of question answering system over knowledge graph.

candidates are used to generate a formal query given the entity and relation mentions to retrieve the corresponding answer from the knowledge graph. Therefore, the goal of shallow parsing is to annotate the words/phrases of the input questions with appropriate labels, which later will be employed in the entity/relation linking step.

The above task resembles a standard classification problem in which every entity/relation mention of the input question is required to be labeled according to its role in fetching the corresponding target entity/relation. However, to the best of our knowledge, almost all existing question-answer datasets do not provide the target annotations for parsing (there is only one such dataset according to [8]). Consequentially, supervised learning approaches (including well-known NLP parsers) fall short in addressing the shallow parsing problems in QA systems over KGs. Nonetheless, since the target entities/relations are the building blocks of the formal query, we can exert the performance of the linker module as a distant measure for evaluation of the parser. For instance, in the given example of Figure 5.1, if the parser only labels "*Barak*" as an entity and not "*Barak Obama*", the linker may not be able to accurately retrieve the corresponding target entity, thus the lower performance of the linker can be interpreted as a distant measure to evaluate the performance of the parser.

Hence, we propose a novel sequential approach to model the task of distant supervision in a Reinforcement Learning (RL) framework where the long-range loss determines a delayed reward signal. Although reinforcement learning has demonstrated successful performance in many complex tasks [73, 74], it has been insufficiently explored in NLP domain. Some examples are using RL in text summarization [75], structure induction [76], and dialog systems [77, 78]. The cognitive process

of human to analyze a question (or a sentence in general) is sequential in nature, and in our case of English language, from left to right. More precisely, while a question is read from left to right, after spotting the first entity/relation mention, identifying the subsequent entities/relations becomes easier by relating the next words to the previously detected parts. Reinforcement learning approaches are able to perfectly replicate this natural behavior. We thus model the task of shallow parsing in QA systems over KGs in a Markov Decision Process (MDP) framework in which the labels can be only obtained at the end of the sequence and are equivalent to the delayed reward in RL scenarios. Therefore, the MDP setting copes with the distantly supervised problem, by utilizing the distant metrics as the reward of a complete question. We further employ a policy gradient optimization technique to learn an optimal policy for labeling the constitutes of the questions in this framework.

Empirically, our model exhibits excellent performance compared to the state-of-the-art methods in the tasks of entity/relation recognition on three popular benchmark datasets. Furthermore, we demonstrate that our approach can be easily integrated into existing entity/relation linking tools and enhances their accuracy, which leads to improving the overall performance of the QA systems.

5.2 Related Work

Traditionally, entity linking and relation linking problems are addressed independently in different applications. There are many works that solely focus on entity recognition and linking [79–84]. These approaches exert a wide range of techniques and methods ranging from heuristics and rule-based methods to supervised learning, and most recently deep neural networks. Shen et al. [85] provide a survey on various approaches and tools for entity disambiguation and linking and their applications in different fields such as question answering. They conclude that the process of entity linking commonly consists of two main steps: generation of candidate entities, in which irrelevant items are filtered out, and a ranking method to identify the best candidates.

On the other hand, several other approaches exist that concentrate only on the task of relation detection as well as linking [86–88]. In contrast to relation extraction that have been an active field of research in natural language processing domain [89–91], relation linking has received less attention. However, relation linking has an important role in various applications e.g., question answering, knowledge graph completion, etc. Legacy methods such as PATTY [88] and BOA [87] provide natural language patterns and taxonomy to create different representations of the relations in the knowledge graph. These approaches are additionally employed in question answering systems [48, 49, 92, 93].

Nevertheless, recent works attempt to exploit the correlation between entities and relations by jointly linking them in the questions in order to improve the performance of the state of the art methods [69, 70]. Dubey et al. [70] propose an approach for entity and relation linking, called EARL, which employs SENNA [94] to extract the entity and relation spans of the questions. They map the task of entity/relation linking to generalized travelling salesman problem, which results in exploiting the connection between entities and relations in the questions. Although SENNA attained the state of the art results in multiple tasks, its performance in QA is inferior as it was trained on long text documents from Wikipedia. Whereas the questions in QA datasets are usually short; for instance the average length of questions in LC-QuAD [95] dataset is 12.3. Falcon [69] is based on a set of manually crafted rules and benefits from general structures of English language to detect the entity/relation mentions. They further merge multiple existing knowledge graphs to increase the lexical diversity of the KG. However, their approach does not generalize well on other datasets.

The usage of RL in various question answering as well as dialog systems has been exploited [30, 96–98]. For instance, Zhong [30] proposed an RL-based model to map questions to SQL query using pointer networks. However, as they need large amount of training data, they used crowd sourcing to generate natural language question for the automatically created queries. A similar work to our research is presented by Takanobu et al. [99] in the NLP domain, where the main objective is to identify the relations. They leverage a two level RL-based framework to first detect a relation in the top level, which triggers the lower level to find the corresponding entities that are semantically associated to the selected relation. Their proposed method is evaluated on two very small datasets (total of 29 and 11 relation). Nevertheless, the approach is presented for scenarios in which the gold-annotations are available which makes it unsuitable to compare to our approach and the reward is computed from the prediction error.

5.3 Distant Supervision Model

In this section, we present our approach for shallow parsing of questions in natural language for QA systems that operate over knowledge graphs. The task is phrased as a reinforcement learning method to model the problem of distant supervision and is thus formulated in a Markov Decision Process (MDP) framework.

5.3.1 Preliminaries

We are given a set of N questions $\mathbb{Q} = \{q_i, \ldots, q_N\}$ and a set of their corresponding formal queries $\mathbb{Z} = \{z_1, \ldots, z_N\}$. Each question $q_i \in \mathbb{Q}$ consists of a sequence of n_i words, $q_i = [\boldsymbol{w}_1^i, \boldsymbol{w}_2^i, \ldots, \boldsymbol{w}_{n_i}^i]$, which are determined with an arbitrary vectorized representation of size d, $\boldsymbol{w}_j^i \in \mathbb{R}^d$. On the other hand, the formal queries are the source of quantifying the so-called distant labels in our setting. A query $z_i \in \mathbb{Z}$, corresponding to q_i, is formed from a set of $m_i \leq n_i$ linked items, $z_i = \{l_1^i, l_2^i, \ldots, l_{m_i}^i\}$, where a linked item l_j^i is defined as a triplet of (*title*, *URI*, *label*) that links a part of the question (one or more words), i.e., *title*, to a *URI* entry in the knowledge graph with a *label* of either *relation* or *entity*. For instance, given the example question of Figure 5.1, the linked items are as follows {(“*almaMater*", `dbp:almaMater`, **relation**), (“*Barack Obama*", `dbr:Barack_Obama`, **entity**), (“*spouse*", `dbp:spouse`, **relation**)}.

We aim to design a parsing method, which receives a question $q_i \in \mathbb{Q}$ and identifies its entities and relations by classifying every word into entity, relation, or none. Therefore, every word $\boldsymbol{w}_j^i$ in the question q_i is assigned to a label $y_j^i \in \{2, 1, 0\}$ which stand for entity, relation, and none, respectively. Consequentially, the output vector of the parser for the question “*What are the schools where Barak Obama’s wife has studied*" is $y^i = [0, 0, 0, 1, 0, 2, 2, 0, 1, 0, 0]$.

5.3.2 The MDP Framework

We model the task of question parsing as a sequential approach based on Reinforcement Learning (RL). The cycle of learning in an RL problem consists of an agent perceiving the state of the environment, performing an action, accordingly, and the environment provides a feedback to evaluate its action. The (delayed) feedback thus operates as a reinforcing signal for optimizing the internal model of the RL framework. An RL problem is mathematically described via Markov Decision Processes (MDPs) [100].

An MDP is represented via a five-tuple $\langle \mathbb{S}, \mathbb{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$, where $\mathbb{S}$ is the state space, $\mathbb{A}$ is the set of actions, $\mathcal{P} : \mathbb{S} \times \mathbb{A} \times \mathbb{S} \rightarrow [0, 1]$ is the transition probability function where $\mathcal{P}(s_t, a_t, s_{t+1})$ indicates the probability of going to s_{t+1} after taking action a_t in state s_t at time t, in which $s_t, s_{t+1} \in \mathbb{S}$ and $a_t \in \mathbb{A}$, $\mathcal{R} : \mathbb{S} \times \mathbb{A} \rightarrow \mathbb{R}$ is the reward of a state-action pair, and $0 < \gamma \leq 1$ is the discount factor. The goal of an MDP is to learn a policy $\pi : \mathbb{S} \times \mathbb{A} \rightarrow [0, 1]$ which maximizes the expected obtained reward. A stochastic policy π gives a probability distribution over the possible actions that the agent can take in the current state.

In our setting, we assume that the input questions are equivalent to the episodes in the reinforcement learning framework. The agent traverses the question from left to the right and decides to choose a label for every word in the sequence based on the information encoded in the current state. At the end of the episode, the obtained labels are integrated and are used to compute a distant loss value which forms a delayed reward signal. In the remainder of this section, we characterize our shallow parsing problem in an MDP framework. Moreover, we discard the superscript i for brevity in the notation and denote the time step within the episodes of the MDP by index t.

States. In our setting, the state space is defined as a subsequence of the question at each time step t as well as the last chosen action. We introduce a parameter h to control the size of state by considering a window of $2h + 1$ words over the input question. A state s_t thus encodes the current word $\boldsymbol{w}_t$, h previous words, h next words, and the previous selected action, $s_t = [\boldsymbol{w}_{t-h}, \ldots, \boldsymbol{w}_t, \ldots, \boldsymbol{w}_{t+h}, a_{t-1}]$. Consequently, taking action a_t at this time step will lead to the next state $s_{t+1} = [\boldsymbol{w}_{t-h+1}, \ldots, \boldsymbol{w}_{t+1}, \ldots, \boldsymbol{w}_{t+h+1}, a_t]$.

Actions. We aim to find the mentions of relations and entities of the questions by classifying their words into a possible set of three labels. We hence specify a discrete action space of $\mathbb{A} = \{0, 1, 2\}$, where $a_t \in \mathbb{A}$ determines whether the current word is an entity or a relation, or it is out of our interest. At the end of the episode (question), the selected actions, $A_q = [a_1, \ldots, a_n]$, form a sequence of mentioned labels for the corresponding words. Note that an action is equivalent to a predicted label, i.e., $a_t = \hat{y}_t$.

Transition function. As choosing an action would only lead to one possible next state, the transition function is deterministic in our problem. That means, $\mathcal{P}(s_t, a_t, s_{t+1}) = 1$ for $s_{t+1} = [\boldsymbol{w}_{t-h+1}, \ldots, \boldsymbol{w}_{t+1}, \ldots, \boldsymbol{w}_{t+h+1}, a_t]$ and is zero otherwise.

Reward function. Since the true labels of the words are not given in every state, we are able to evaluate the selection policy only at the end of the question. Therefore, no immediate reward is available at each time step. We thus delay the policy evaluation and utilize the assessment of the linker to compute a distant score for our prediction which serves as a delayed reward for updating the policy. We describe our method for computing the distant feedback below.

5.3.3 The Distant Labels and Reward

Consider the question provided in Figure 5.1 again. The example depicts certain complexities in the parsing task that we should take into account. An entity or a relation could refer to more than one word. For instance, the entity `dbr:Barack_Obama` is specified by separately labeling two words *Barak* and

Obama as entity. Furthermore, as the example illustrates, there is not always a one-to-one mapping between a linked item and a word (or set of words) in the question, e.g., *wife* vs. `dbp:spouse` and *schools* vs. `dbp:almaMater`. In addition, recall that the true labels, i.e., entity/relation mentions in the questions, are not available in our problem.

We propose to learn a policy for labeling every word $\boldsymbol{w}_t$ of a question q from a distant reward value which is computed from the quality of linked items $l \in z$. As a result, our RL framework becomes a distantly supervised approach for the underlying parsing task. To do so, we find the phrases (sequence of words) from the question q which provide the best indication for each target linked item l_j. Let a_t^π be the action (aka label) chosen by policy π for word $\boldsymbol{w}_t$ and assume that a_t^π is either entity or relation, i.e., $a_t^\pi \neq 0$, we group the words within the same label to construct the set of entity and relation mentions for the question at-hand. For instance, if $(a_{t-1}^\pi \neq a_t^\pi) \wedge (a_t^\pi = a_{t+1}^\pi = \cdots = a_{t+b}^\pi) \wedge (a_{t+b}^\pi \neq a_{t+b+1}^\pi)$, we concatenate these sequence of b words into one phrase, $\omega_k = [\boldsymbol{w}_t, \ldots, \boldsymbol{w}_{t+b}]$ with label $\lambda_k = a_t^\pi$. Hence, at the end of the episode (question), a set of g entity and relation mentions $\Omega = \{\omega_1, \ldots, \omega_g\}$ are obtained along with their predicted labels $\Lambda = \{\lambda_1, \ldots, \lambda_g\}$, where $g \leq n$, and optimally $g = m$.

Using a well-qualified similarity function $\phi(.,.) \mapsto \mathbb{R}$, we aim to compute a score for prediction of each (ω_k, λ_k) pair from the actual policy by finding a linked item from z, which is of the same label as λ_k and its title is the most similar to ω_k

$$\text{score}(\omega_k, \lambda_k) = \max_{l \in z,\ \text{label}(l) = \lambda_k} \phi(\omega_k, \text{title}(l)). \tag{5.1}$$

As a result, $\text{score}(\omega_k, \lambda_k)$ computes how relevant the predicted label for the phrase ω_k is to the label of the most similar target linked item, and will later be used to define the delayed reward of an episode/question. Continuing with the question in the earlier example, if the model labels the word *"where"* as a relation, it would get a very low score as it is not similar to neither of both target relations. In contrast, if the word *"Obama"* is marked as an entity, it will be matched to `dbr:Barack_Obama` with a fairly high score, and when the model correctly identifies both *"Barak"* and *"Obama"* as an entity phrase, we get an almost exact match. Note that the choice of similarity function is very essential as it should be able to detect the words with the same meaning, such as *"spouse"* and *"wife"*, as well as typographical errors, for instance the letter *"c"* is missing in *"Barak"*.

Now we utilize the obtained scores to define a distant feedback for the whole episode which provides a delayed reward. Given the scores computed from Equation (5.1), the total reward is defined as the average scores of all identified phrases in the question

$$r = \frac{1}{g} \sum_{k=1}^{g} \text{score}(\omega_k, \lambda_k), \tag{5.2}$$

where r also represents the so-called distant measure for the whole question. The obtained reward is further discounted by the factor γ for the previous states and is used to specify feedback for the overall episode, particularly, the words that are not identified as any entity/relation mention and labeled as zero.

5.3.4 Optimization

Once the shallow parsing task is successfully phrased in an MDP framework, we aim to learn a deterministic policy π which provides the best action (annotation) given the actual state (word). Note

that the learned policy in our RL framework is a stochastic policy, and we can simply turn that into a deterministic action selection using $\pi(s) = \arg\max_a \pi(a|s)$.

Subsequently, we take a policy gradient method [101] to directly learn the optimal policy without learning the intermediate value functions. The goal of policy gradient algorithm is to learn a policy π with parameters θ, i.e., π_θ, where following that policy maximizes the "expected" obtained reward. Recall that in our setting, the episodes are formed from the available questions in the training data. However, following various policies creates different episodes from a single question which leads to a total of M episodes. We thus assume that an episode τ_i is formed when following policy π_θ with a probability of

$$P(\tau_i; \pi_\theta) = P(s_1^i) \prod_t \pi_\theta(a_t^i|s_t^i)\mathcal{P}(s_t^i, a_t^i, s_{t+1}^i),$$

and since the transition probability is deterministic, we have

$$P(\tau_i; \pi_\theta) = P(s_1^i) \prod_t \pi_\theta(a_t^i|s_t^i).$$

Let r^i be obtained from Equation (5.2) and $R(\tau_i) = \sum_t \gamma^{t-1} r_t^i$ be the total discounted reward acquired for episode τ_i while following policy π_θ, the objective function is defined as

$$J(\theta) = \sum_{i=1}^{M} P(\tau_i; \pi_\theta) R(\tau_i), \tag{5.3}$$

which aims to maximize the expected reward over all possible episodes, weighted by their probabilities under policy π_θ. Hence, we estimate the gradients in the direction of higher discounted reward to update the parameters θ of the policy π via gradient ascent

$$\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta),$$

where α is the learning rate, and by taking the log of likelihood, the gradient becomes

$$\nabla_\theta J(\theta) = \sum_{i=1}^{M} \nabla_\theta \log \pi_\theta(a_t^i|s_t^i).$$

Given the objective function in Equation (5.3), we employ a deep learning method as a general function approximation technique to learn the parameters θ. We thus design a fully connected neural network with three layers as the policy network. In the first layer, the words are vectorized using a word vectorization technique e.g., word embedding [102, 103]. The state is then created using the current, the h previous and h next vectorized words along with the last chosen action, which is a numerical value. The second layer can be either a linear transformation with Relu as an activation function, or an LSTM or a Bi-LSTM. We compare different architectures in Section 5.4. The output layer uses Softmax activation to provide the final action distributions, from which the agent would sample the next action $a_{t+1} \sim \pi_\theta(s_t, a_t)$.

5.4 Empirical Study

In this section, we evaluate the performance of our proposed approach for shallow parsing in QA systems over KGs. We conduct our experiments on three popular benchmarking datasets: (i) LC-QuAD [95], (ii) QALD-6 [104], and (iii) QALD-7 [105]. The first dataset consists of 5,000 manually crafted question-query pairs that covers a wide range of vocabulary, semantic complexity, spelling and grammar mistakes, and types of question. The other two datasets are from the Question Answering over Linked Data (QALD [1]) challenge series. QALD-6 contains 350 training questions as well as 100 questions in the test set, whereas QALD-7 has 215 training questions and 50 questions in the test set.

Recall that in our distantly supervised setting, the target entity/relation mentions are not available. Therefore, we evaluate our parser in combination with a linker to the KGs as a proxy for assessing its performance compared to the baseline linking methods. In the remainder, we denote the combination of our parser and the linking component by *MDP-Parser*. The performance of different methods is evaluated via *accuracy* which indicates whether the top candidate obtained from our method correctly matches the items identified in the target query. Furthermore, we employ *Mean Reciprocal Rank (MRR)* in cases where a list of candidates with length k are presented MRR $= \frac{1}{k}\sum_{i=1}^{k}\frac{1}{\text{rank}_i}$, where rank_i specifies the position of the target item in the list of candidates. In addition, we employ a Bayesian optimization approach using SigOpt [2] [106] for hyper-parameter optimization on the validation set.

Reproducibility. The source code and all intermediate files of our method is available on `https://github.com/AskNowQA/DeepShallowParsingQA` to facilitate reproducibility.

5.4.1 Linking Component

We design a simple linking component which is able to retrieve the entities and relations from the knowledge graph given the output of the shallow parsing. The linking component consists of two steps: retrieving the candidate items and ranking them. In the first step, we employ Apache Lucene [3] to create a full-text index of the character-level tri-grams of the entities/relations from the underlying KG. These obtained indices provide a list of candidate URIs given the mentions that parser predicted to be either an entity or a relation (see Figure 5.2). Note that we distinguish *"relation"* linking from *"entity"* linking in the evaluation as they have distinct characteristics. Furthermore, in order to increase the accuracy of the relation linking, we extend the output of the relation index with the relations that are in a one-hop distance of the top candidate entity for each entity mentions. In the second step, the candidates are ranked according to a pair-wise ranking function which utilizes an appropriate similarity method.

We consider three different techniques for computing the similarity in the ranking function that capture either the semantic similarity, surface resemblances, or both. In order to capture the string similarity, we examine the top-10 best performing techniques for full name matching reported by [107], and we choose levenshtein distance (LEV) to capture the string similarity of a given mention and the corresponding candidates, which performs the best in our setting. Second, to support semantic similarity, we utilize GloVe vectors [102] to initialize the word embedding (EMB) vectors, where we decide for cosine similarity over euclidean distance and word mover's distance [108] due to its efficiency in terms of performance and time.

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

[2] http://www.sigopt.com

[3] https://lucene.apache.org

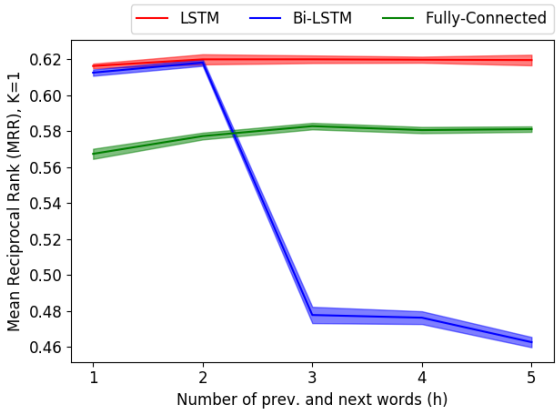


Figure 5.3: Performance in terms of MRR for various network architectures w.r.t. state size on LC-QuAD dataset.

Finally, we combine both of them via a linear function (LEV+EMB). In the experiments, we use LEV for the entity linking and LEV+EMB for the relation linking, which intuitively means that the string similarity is sufficient for matching the entities, while for the relations semantic similarity is also required.

5.4.2 Baseline Methods

We evaluate the performance of our approach in comparison to four categories of baseline methods. First, we employ two well-known NLP parsers, namely SENNA [94] and Flair [72], combined with our linking component (see Section 5.4.1) to measure the gain in the performance of our parser, which is designed for QA systems, compared to an existing NLP shallow parser. Note that, to the best of our knowledge, no parser which is specifically designed for QA systems is available to be used as other baselines. Second, we compare our approach to an LSTM as well as a Bi-LSTM model which converts the obtained scores into weak labels to address the parsing problem as a classification task. This helps to highlight the importance of modeling the distant supervision problem in an RL paradigm in contrast to using local scores.

The third group of baselines includes the state of the art entity/relation linking methods that already contain an internal parser. We take EARL [48] and Falcon [69] that report results on both entity as well as relation linking tasks. In addition, Babelfy [82], FOX [83], TagMe [79] and DBpedia Spotlight [81] serve as baselines for entity linking, and SIBKB [109] and ReMatch [86] are used for the task of relation linking. Finally, in order to show the effectiveness of our approach in combination with existing entity/relation tools, we substitute the shallow parsing component in EARL with our parser and address this baseline as *EARL+MDP-Parser*. In general, our parser could be re-used in different methods and combined with various linkers to form a powerful pipeline.

Table 5.1: Accuracies for *entity* linking task.

Approach	LC-QuAD	QALD-6	QALD-7
SENNA	0.27	0.22	0.22
Flair	0.59	0.56	0.43
LSTM	0.47	0.43	0.46
Bi-LSTM	0.39	0.25	0.15
Babelfy	0.17	0.29	0.29
FOX	0.40	0.49	0.41
TagMe	0.30	0.34	0.41
EARL	0.60	0.50	0.56
DBpedia Spotlight	0.54	0.68	0.62
Falcon	0.74	**0.70**	0.38
MDP-Parser	**0.76**	**0.70**	**0.69**
EARL+MDP-Parser	0.71	0.61	0.62

5.4.3 Performance Results

Model Selection

In the first experiment, we analyze the impact of two of the main design factors of our model: the state size which is controlled by parameter h encoding the number of previous and next words included in the state, and the architecture of the deep model.

Figure 5.3 reveals the effect of different choices of models on the performance of entity and relation linking task in terms of MRR where $k = 1$. The experiments are repeated for several runs and report the average results with standard errors. The figure illustrates that the LSTM model gives better results than the other models regardless of the value of parameter h. Additionally, the state size of 3, i.e. $h = 2$, achieves the best performance in different deep models, however comparing to $h = 1$, the improvement is marginal, but the computational costs are much higher. Hence, in the remainder of this section, we set $h = 1$ and use the deep architecture that contains an LSTM layer for our empirical study.

Evaluation of Entity/Relation Linking

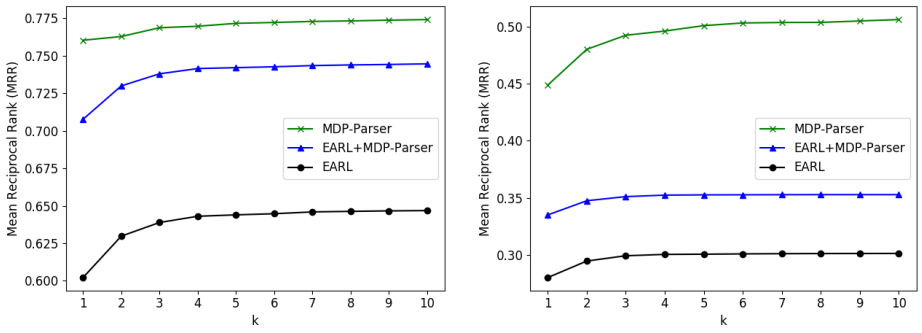
We first evaluate the performance of various approaches for the task of entity linking. Table 5.1 presents the obtained results on the test-set of LC-QuAD, QALD-6 and QALD-7 datasets. The table demonstrates that our MDP-based method outperforms different baseline systems in LC-QuAD and QALD-7 datasets. Although Falcon has a good performance on two datasets, it has significantly lower performance on QALD-7 data[4]. The reason for this lies in the fact that Falcon is a rule-based system which requires manual rule-extraction, and the authors reportedly used other data to extract the rules. The low accuracy of 38% leads to the conclusion that the extracted rules are overfitted and do not generalize well to other datasets. Additionally, looking closer to the result of EARL+MDP-Parser illustrates that our method significantly increases the performance of EARL by more than 10% on LC-QuAD as well as QALD-6, and by a factor of 6% on QALD-7 dataset.

[4]The results of Falcon are computed via: https://labs.tib.eu/falcon

Table 5.2: Accuracies for *relation* linking task.

Approach	LC-QuAD	QALD-6	QALD-7
SENNA	0.10	0.02	0.06
LSTM	0.28	0.20	0.15
Bi-LSTM	0.25	0.06	0.03
SIBKB*	0.14	N/A	N/A
ReMatch*	0.16	N/A	N/A
EARL	0.26	0.18	0.15
Falcon	0.38	0.30	0.13
MDP-Parser	**0.45**	**0.34**	**0.25**
EARL+MDP-Parser	0.34	0.22	0.24

* Results are taken from [69]



(a) Entity linking

(b) Relation linking

Figure 5.4: Performance in terms of MRR w.r.t. the value of k.

We further evaluate the performance of our method compared to the baselines for the task of relation linking. Table 5.2 summarizes the results in terms of accuracy. Despite the simplicity of the linking component in our approach, MDP-Parser achieves better results compared to the baselines, and again, improves the performance of EARL by almost 5% on all the datasets. Note that the overall low performance of relation linking methods in comparison to the entity linking is due to the intrinsic complexity in various dimensions, such as difficulty of capturing the relation mentions, semantic closeness, similar candidates from the underlying KG, etc.

Moreover, we analyze the performance of the approach for retrieving top-k candidates instead of only one. In the real scenarios, by integrating a linker into a full question answering pipeline, the query generator can process multiple candidates per entity/relation mention (see [110]), which leads to increase in the performance of the whole QA pipeline (see Section 5.4.3). Hence, we expand the list of generated candidates to consider top-k items and report the performance in terms of MRR in Figure 5.4 for EARL, MDP-Parser and EARL+MDP-Parser (Falcon is omitted as it provides a single candidate per each entity/relation mention). The results show that MDP-Parser constantly outperforms EARL for different values of k in both entity linking (a) and relation linking (b) tasks. Particularly, we observe a significant enhancement (about 5%) of our approach in relation linking, which is due to the fact that there are linked items with a very similar label (e.g., *dbr:school* and *dbr:schools*) in the KG.

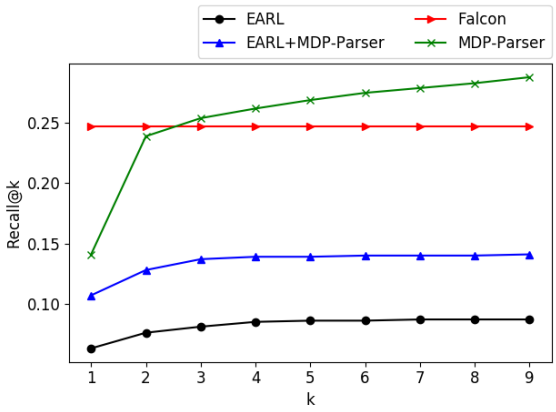


Figure 5.5: The recall of various QA pipelines w.r.t. the value of k.

This means that by providing more items in the list of candidates, the query generator is at liberty to use the most appropriate candidate.

Performance of an Overall QA Pipeline

In order to study the impact of our shallow parsing approach in a complete question answering pipeline, we implement the pipeline shown in Figure 5.2, in which we employ SQG [110] to serve as the query building component. In this pipeline, the parser annotates the input questions as required by the linker component. The linker component further extracts a list of candidates from the underlying KG for each input from the parser. Lastly, the query builder component takes the candidate items from the linker and generates the final candidate formal queries.

We thus instantiate four realizations of the pipeline: (i) EARL with its internal parsing (ii) Falcon with its internal parsing (iii) combination of MDP-Parser with our ad-hoc linking component explained in Section 5.4.1 (iv) EARL as the linking component with MDP-Parser as parsing component. Figure 5.5 depicts the performance of each pipeline on the LC-QuAD dataset in terms of recall, which is defined as whether the query builder was able to create the target formal query using top-k candidates by the linker, irrespective of its ranking among all candidate formal queries. We can observe from Figure 5.5 that the pipeline using MDP-Parser, i.e. (iii), achieves the best score for $k \geq 3$. Also, the pipeline that uses MDP-Parser as the shallow parser and EARL as linker (EARL+MDP-Parser) achieves a performance boost of about 5% in comparison to EARL.

Case Sensitivity

In many scenarios, the input questions from the users contain different variety of mistakes including ignoring capitalization. The lower/upper case usually helps to find out the entities, and many approaches depend on them. In this experiment, we show that our approach is not affected as much as others. We thus evaluate the robustness of our method, with respect to case sensitivity, compared to two state of the art baselines: EARL and Falcon. To do so, we modify all the questions in the LC-QuAD data to lowercase and re-evaluate the methods. Table 5.3 shows that the performance of both baselines remarkably falls down on the modified dataset. The accuracy of EARL drops by 60% and 10%, and Falcon by 29%

Table 5.3: Accuracies on LC-QuAD in lowercase

Approach	Entity linking	Relation linking
EARL	0.006	0.164
Falcon	0.453	0.152
MDP-Parser	**0.612**	**0.443**

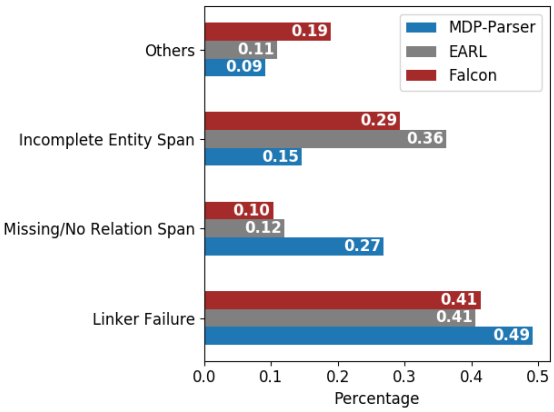


Figure 5.6: Error analysis of the results.

and 23% for entity linking and relation linking tasks, respectively (cmp. Table 5.1 and Table5.2). On the other hand, MDP-Parser performs almost the same as before for relation linking (only 1% decrease). However, the accuracy of entity linking drops by 15%. This result indicates that while MDP-Parser certainly benefits from capitalization, it does not totally depend on that compared to the baselines.

5.4.4 Error Analysis

To better understand the success and failure cases of MDP-Parser and the baseline systems, we randomly select 250 out of 1,000 questions from the LC-QuAD test set and perform a thorough manual assessment of failure cases for MDP-Parser, EARL, and Falcon in four categories: Linker Failure, Missing/No Relation Span, Incomplete Entity Span, and others. Figure 5.6 depicts the failure percentage of different methods in these four groups. The most failure in all three methods occurs because of failure in the linking component, when the linker fails to retrieve the target items, given that the shallow parser provided the correct entity/relation mentions. For instance, in the question `To which places do the flights go by airlines headquartered in the UK`, the phrase `UK` was identified as a potential entity, yet the linker failed to find `dbr:United_Kingdom`.

The second most common case for MDP-Parser is where it fails to extract at least one relation in the question. As an example, given the question `Who are the stockholder of the road tunnels operated by the Massachusetts Department of Transportation?`, with target relation set of `[dbp:owner, dbp:operator, dbo:RoadTunnel]`, MDP-Parser correctly identified `stockholder` and `operated` as candidate relation spans, however, failed to recognize `road tunnels`. However, the baseline systems tend to offer more relation spans than MDP-Parser, which caused them to falsely suggest the relation mentions that are not related to the target relations.

The next frequent reason for failure in MDP-Parser is incomplete entity span, which usually occurs for long entity mentions. As an instance, for question `Who created the Women in the Garden and also the L'Enfant a la tasse?`, MDP-Parser correctly identified `L'Enfant a la tasse` as one entity, however, failed to cast `Women in the Garden` as an entity mention, instead it presented `Woman` and `garden` as two separated entity mentions. Still, MDP-Parser generally handles long entity mentions better than the baseline systems. Lastly, there are less frequent cases such as the instances where two entity/relation mentions reported as one, or where there is no entity mention, etc.

5.5 Conclusions

In this chapter, we designed a shallow parsing method, called MDP-Parser, for question answering systems over knowledge graphs. Considering that the true labels are not available, we presented a sequential approach based on reinforcement learning which is able to model the distantly supervised task by computing a delayed reward signal from distantly computed scores. Furthermore, we combined our parser with an ad-hoc linking component and empirically showed that our method significantly outperforms the baseline systems. In addition, integrating our method into an existing linker remarkably boosted the performance of the linker on the benchmarking datasets.

In the next chapter, the focus is still on the **RQ-2**, though on query building task of question answering pipeline. We present *SQG*, a semantic query generator, that generates candidate formal query. Assuming a natural language question and a set of candidate entities and relations, SQG generates formal queries that form a valid walk within the underlying knowledge graph. It further ranks them based on the structural similarity in comparison with the structure of input question in order to find the one that capture the correct intention of the input question.

CHAPTER 6

Query Builder

In the previous chapter, we discussed the shallow parsing task in detail as part of the contribution toward **RQ-1**. Due to the lack of target labels for the task, supervised methods fell short, thus we devised a reinforcement learning based model to learn the parameters based on the distance labels. We showed that our model led to a remarkable increase in the performance of the entity and relation disambiguation step as well as the overall performance of the question answering system.

In this chapter, we assume that the first two tasks in semantic question answering, namely shallow parsing and entity/relation disambiguation, are already carried out and we focus on the query building task. In the following section, we provide the motivation for the query building task, and enumerate its challenges. In Section 6.2, we review the existing works on query building approaches as well as the query ranking methods. Section 6.3 first elaborate an algorithm to generate the candidate formal queries given the set of linked entities and relations. Also, it describes a neural network based model to rank the candidate queries by exploiting the structural similarity of the input question and the candidate queries. Section 6.4 provide empirical study and Section 6.5 summarize our findings as to approach the second research question.

6.1 Introduction

Query building is an integral part of any semantic question answering system. However, in most semantic question answering, the query building task is mixed with task in the pipeline which results in the general inability by the wider community to successfully and efficiently build upon the efforts of past achievements [10]. To tackle this problem, researchers introduced QA modular frameworks for reusable components such as OKBQA [9, 11] but little attention has been given to query generation. For instance, OKBQA includes 24 reusable QA components of which only one is a query generator. Nonetheless, the increasing complexity of questions leads to the following challenges for the query generation task [10, 14]:

1. Coping with large-scale knowledge bases: Due the very large size of existing open-domain knowledge bases such as DBpedia [111] and Freebase [17], special consideration is required.

2. Question type identification: For instance the question might be a boolean one, thus the query construction should be carried out accordingly to generate desired answer.

3. Managing noisy annotations: The capability to handle a set of annotations including several incorrect ones might increase the chance of QG to construct the correct query.

4. Support for more complex question which requires specific query features such as aggregation, sort as well as comparison.

5. Syntactic ambiguity of the input question: For example, "Who is the father of X?" might be interpreted as "X is the father of who?" if the syntactical structure of the question is ignored.

To the best of our knowledge, none of the existing works can deal with all these challenges. Thereby, we present SPARQL Query Generator (SQG), a modular query builder for QA pipelines which goes beyond the state-of-the-art. SQG is able to process noisy input and employs a ranking mechanism of candidate queries based on Tree-LSTM. The model overcomes the limitations of linear solutions [112, 113] by exploiting the structural similarity of the candidate queries with the dependency of the utterance. We also considered scalability aspects, which enables us to evaluate SGQ on a large Q/A dataset based on DBpedia [111].

6.2 Related Work

Diefenbach et al. [10] studied more than 25 QA systems and discussed the different techniques these used for each component, including the query generation component. They show that in most QA systems the query generation component is highly mixed with components performing other tasks, such as segmentation or relation extraction. Furthermore, their work mainly focused the analysis on the overall performance of the QA systems, as only a few systems and publications provided deep analysis of the performance of their query generation component. For instance, CASIA [114], AskNow [48] and Sina [115] each have a SPARQL generation module, but do not provide an evaluation on it. [14] is a very comprehensive survey of question answering over knowledge graphs, analyzing 72 publications and describing 62 question answering systems for RDF knowledge graphs. In particular, they argue that it may not be beneficial that a wide range of research articles and prototypes repeatedly attempt to solve the same challenges: better performance may be obtained by providing sophisticated and mature solutions for individual challenges. An alternative approach to QA pipelines is end-to-end systems such as [116], which directly construct SPARQL queries from training data. However, those are currently mostly restricted to simple questions with a single relation and entity in which case query generation is straightforward. While those systems are an interesting area of research and may be extensible to more complex questions, they are unlikely to completely replace QA pipelines due to the large amount of training data required.

To the best of our knowledge, there is a very limited number of working query builders in the question answering community which can be used independently and out of the box. Recent studies [9, 11] explored the possibility of using independent QA components to form a fully functional QA system. Singh et al. [9] introduced Frankenstein, a QA framework which can dynamically choose QA components to create a complete QA pipeline, depending on the features of the input question. The framework includes two query building components: Sina and NLIWOD QB. NLIWOD is a simple, template-based QB, which does not include any kind of query ranking. Given correct inputs, NLIWOD compares the pattern based on a number of input entity and relation to build the output query. Sina [115] was originally a monolithic QA system, which did not provide a reusable query builder.

However, Singh et al. [9] decoupled the query building component to make it reusable. Its approach consists in the generation of minimal sets of triple patterns containing all resources identified in the question and select the most probable pattern, minimizing the number of triples and of free variables. Another approach not far from ours is presented by Abujabal et al. [113]. In contrast to approaches such as [52, 112, 117] which rely on handcrafted utterance-query templates, they provided a learning model to generate the the template set. Furthermore, [112, 113] rely on ranking methodologies for choosing among candidate queries. However they rely on question-query templates, previously learned by distant supervision, which are ranked on several manually engineered features by a preference function learned via random forest classification. By contrast, our ranking approach is based on the similarity between candidate walks in a minimal subgraph and the sentence structure of question.

6.3 Approach

We formally describe knowledge graphs and the query generation problem as a subproblem for question answering over knowledge graphs. Within the scope of this chapter, we define a *knowledge graph* as a labelled directed multi-graph: a tuple $K = (E, R, T)$ where E is a set called entities, R is a set of relation labels and $T \subseteq V \times R \times V$ is a set of ordered triples. The definition captures the basic structure of, e.g., RDF datasets and property graphs.

We define *query generation* as follows: We assume that a question string s and a knowledge graph K are given. In previous steps of the QA pipeline, entity and relation linking have already been performed, i.e. we have a set of mappings M from substrings (utterances) of s to elements of E and R, respectively. The task of query generation is to use s, D and M to generate a SPARQL query.

This definition completely decouples the query generation task from the NED and RE tasks. However, in a realistic setting, the NED and RE modules produce a list of candidates per utterance in the question. As a result, we relax the constraint on the utterance annotation to have a list of candidates per each utterance. Formally, we define the task of *advanced query generation* based on the definition of query generation, where each substring of s is mapped to a non-empty subset of E or R respectively, i.e. there are several candidates for entities and relations. For instance, Figure 6.1 illustrates that for the question "What are some artists on the show whose opening theme is Send It On?", the annotation for the "artists" might be *dbo:Artist* or *dbo:artists* and so on. In Section 6.4.2, we show that considering multiple candidates leads to a better performance, in comparison to the case where only the top candidate is considered by the query generator.

Our hypothesis is that the formal interpretation of the question is a walk w in the KG which only contains the target entities E and relations R of the input question s plus the answer nodes. The definition of a (valid) walk is as follows:

Definition 1 (Walk) *A walk in a knowledge graph $K = (E, R, T)$ is a sequence of edges along the nodes they connect: $W = (e_0, r_0, e_1, r_1, e_2, \ldots, e_{k-1}, r_{k-1}, e_k)$ with $(e_i, r_i, e_{i+1}) \in T$ for $0 \leq i \leq k-1$.*

Definition 2 (Valid Walk) *A walk W is valid with respect to a set of entities E' and relations R', if and only if it contains all of them, i.e. :*

$$\forall e \in E' : e \in W \text{ and } \forall r \in R' : r \in W$$

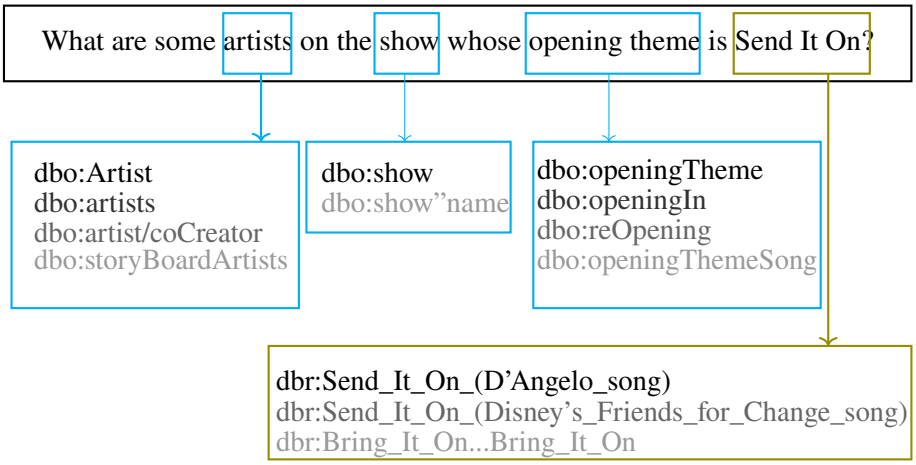


Figure 6.1: A sample question annotated with output from NED and RE components. There is a list of candidates for each spotted utterance in the question ranked based on their confidence score.

A node e with $e \in W$ and $e \notin E'$ is *unbound*. An *unbound node* is an abstract node which is used to connect the other nodes in the walk. This node, however, can be related later to one or a set of specific nodes in the KG.

We carry out the task of capturing the valid walks in two steps: First, we establish a type for the question (e.g. boolean or count). Depending on the type, a number of valid walks are extracted from the KG, however, most of them may be an incorrect mapping of the input question, in the sense that they do not capture the correct intention behind the question. Thus, we sort the candidate walks with respect to their similarity to the input question. The overall architecture of SQG is shown in Figure 6.2 and in the following sections we discuss each step in more detail.

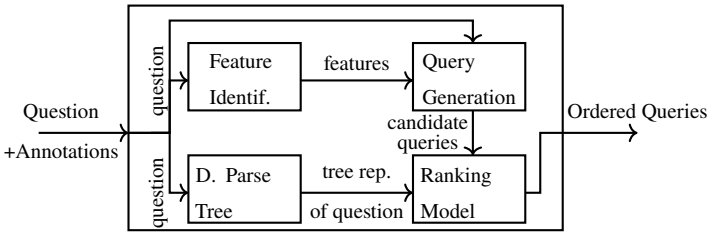


Figure 6.2: The Architecture of SQG

6.3.1 Query Generation

In order to find candidate walks in the KG, we need to start from a linked entity $e \in E$ and traverse the KG. Given the size of the existing KGs such as DBpedia [1] or Freebase [17], however, it is very time-consuming to enumerate all valid walks. Thus, we restrict to the subgraph consisting of all the linked entities and relations. In this subgraph, we can then enumerate the candidate walks, which can be directly mapped to SPARQL queries. Yet, the type of question is a crucial feature that has to be identified in order to create the candidate queries with correct structure from the valid walks.

Capture the Subgraph

We start with an empty subgraph, which is populated with the linked entities E as its nodes. Then, we augment the nodes with edges that correspond to the linked relations R, if such connections exist in the KG. This is illustrated in Figure 6.3, if only the solid lines are considered. In this step, we consider every possible direct way of connecting the entity(s) and the relation(s) to enrich the subgraph: a relation might connect two existing nodes in the subgraph, or it may connect an entity to a new unbound node. Thus, this subgraph might contain several valid walks but the correct one, since the intention of the question might require to include nodes in the two-hop distance. For instance, in Figure 6.3, the answer node ("$unbound_1$") is in the two-hop distance from the entity "dbr:Send_it_on" and is not included in the current subgraph. Consequently, we need to expand the subgraph with the set of candidate relations R. Depending on the question, such expansion might correspond to a very large portion of the underlying KG, which may not be even useful to create the final list of candidate walks. Instead of expanding the subgraph, we expand the existing edges of the subgraph with the set of candidate relations excluding the relation that the existing edge represents. As a result, the search space in the underlying KG is greatly reduced (see Figure 6.3, where the dashed lines represent the edge expansions). Algorithm 1 summarize the process of capturing the subgraph.

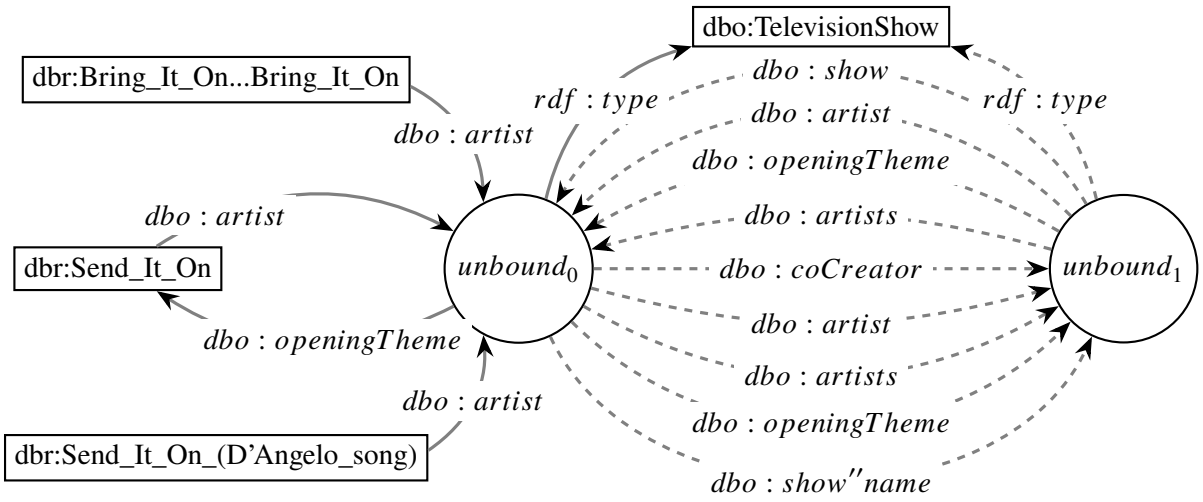


Figure 6.3: The captured subgraph for the given question, annotated with candidate entities and relations; solid lines are the one that are in one hop distance; dashed line means more than one hop distance; circles represent unbound nodes, rectangles are the linked entities and edges are the relations in the KG.

Enumerate Candidate walks

At this point, we have a subgraph that covers all the entities and the relations in the question. We consider every unbound node as a potential answer node, therefore we look for valid walks according to Definition 2. For our example, Figure 6.4 reveals four valid walks. In case there is only one valid walk in the subgraph, we map the walk to a SPARQL query and report it as the corresponding query for the given question. Note that further augmentations might be required to support different types of questions, such as those requiring counts or returning boolean values. If there is more than one valid

Data : E', R', K
Result : G: Minimal covering subgraph
Initialize G as an empty graph;
Add $\forall e \in E'$ to G as nodes;
foreach $e \in E', r \in R'$ **do**
  if $(e, r, ?) \in K$ **then**
    add $(e, r, ?)$ to G ;
  else if $(?, r, e) \in K$ **then**
    add $(?, r, e)$ to G ;
end
foreach $(e_1, r, e_2) \in G$ **do**
  foreach $r' \in R', r' \neq r$ **do**
    if $(e_2, r', ?) \in K$ **then**
      add $(e_2, r', ?)$ to G ;
    else if $(?, r', e_2) \in K$ **then**
      add $(?, r', e_2)$ to G ;
    else if $(e_1, r', ?) \in K$ **then**
      add $(e_1, r', ?)$ to G ;
    else if $(?, r', e_1) \in K$ **then**
      add $(?, r', e_1)$ to G ;
  end
end

Algorithmus 1 : Capture the subgraph

walk, then ranking (as described below) needs to be performed in order to find the most similar query to the input question.

Feature Identification

SQG supports simple and compound questions. In order to support questions such as boolean and count questions, we first need to identify the type of question. Furthermore, there are questions where a relation is required to be used twice. For instance in question *What awards have been awarded both to Ingmar bergman and James O'Brien?* the relation *award* needs to be used twice in order to create the desired query. In some works, predefined patterns were used for similar purposes [48, 118, 119]. However, we train an *SVM* and *Naive Bayes* model for each tasks: one to classify the questions into boolean, count or list questions; and one to decide whether to use a relation twice; based on their *TF-IDF* representation.

Although TF-IDF is a simple representation, the results in Section 6.4.2 reveal that it is strong enough for the purpose. Given the class of the question, the query generator will format the query accordingly. In the case of a count query, for instance, the query generator adds the corresponding function to the output variable of the SPARQL query. Moreover, Based on the predication of the classifier to use a relation twice, an artificial relation will be added to the input of query generator which contains all the candidate relations.

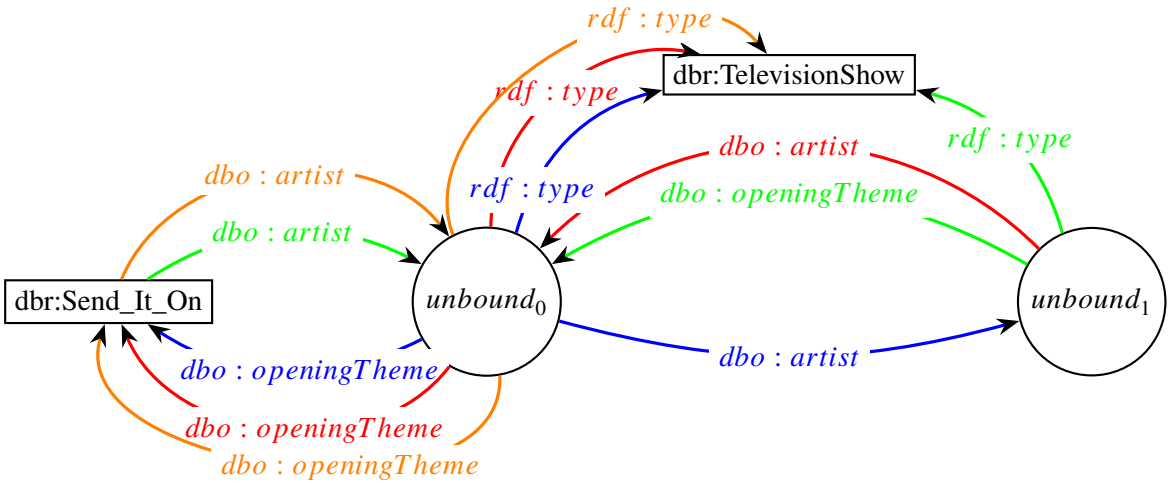


Figure 6.4: Four candidate walks are found which are shown in different colors

6.3.2 Query Ranking

There might be more than one valid walk, due to the uncertainty in the linked entities/relations and complexity of the KGs. As a result, we need a way to rank them with respect to the intention of the input question. Yit et al. [118] proposed a model based on convolutional neural networks to represent the input questions and the core-chains in a latent semantic space and compute the cosine similarity. Although the order of the words is captured to some extent in the model, the overall structure of the input question and candidate core-chains is not taken into account. Considering the fact that the walks consist of many shared entities/relations, our hypothesis for the ranking model is that the structure of the walks is a distinctive feature to distinguish the similarity between the candidate walks and the input question. For instance, four walks are generated for our running example, which have distinct structures (see Figure 6.6). Therefore, the desired model should be able to incorporate the structure of the input. Tai et al. [120] presented Tree-LSTM (Fig. 6.7), an enhanced version of the vanilla LSTM, which considers the tree representation of the input rather than just the sequence of input tokens. They applied the model to the semantic relatedness of natural language sentences and sentiment classification. We present a ranking model based on Tree-LSTM. It considers the tree representation of the candidate walks with respect to the syntactical structure of the question in order to compute their similarity. In the following, we shortly introduce LSTM and Tree-LSTM, subsequently we discuss the final model to fulfill the task.

Preliminaries

As opposed to the vanilla neural networks, where the inputs at different steps are assumed to be independent, Recurrent Neural Networks (RNN) have a hidden state h_t at each step t, which depends not only on the current input but also the previous hidden state h_{t-1}. This formulation enables RNNs to memorize the previous computations for an arbitrarily long sequence of inputs. In practice, however,

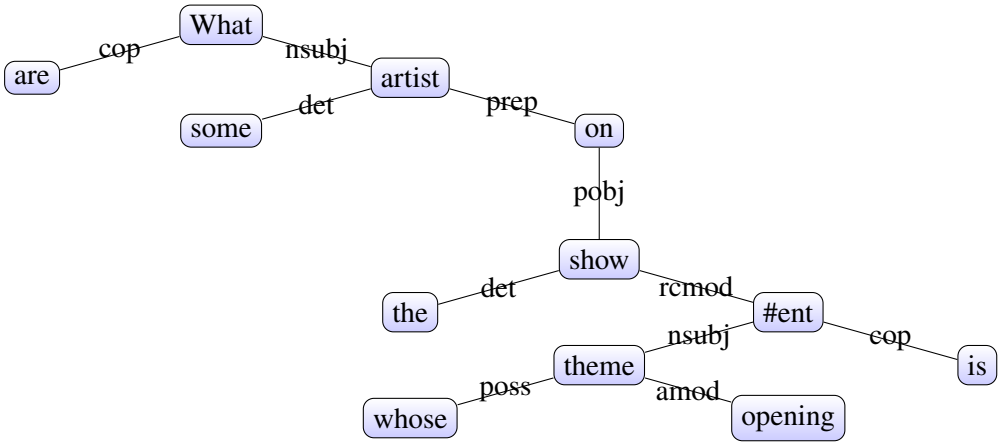


Figure 6.5: Dependency parse tree of the running example

it has been pointed out that the memory functionality of the RNNs is limited to a few steps back [121]. As a remedy to this issue, RNNs with Long-Short Term Memory (LSTM) units were introduced [122]. They are empowered by a memory cell which is able to keep the state for a longer period of time.

Tree-LSTM RNNs and LSTMs consume the input in a sequential manner. However, when the structure of the input is not simply sequential, special treatment is required. Tai et al. altered the architecture of LSTM to support tree-structured input [120]. Tree-LSTMs aim to incorporate information that rests in the child nodes, whilst LSTM support only sequential propagation. Tree-LSTM units take into account the state of its child nodes to compute its internal state and the output. This architecture enables Tree-LSTMs to easily incorporate the tree structure of our input.

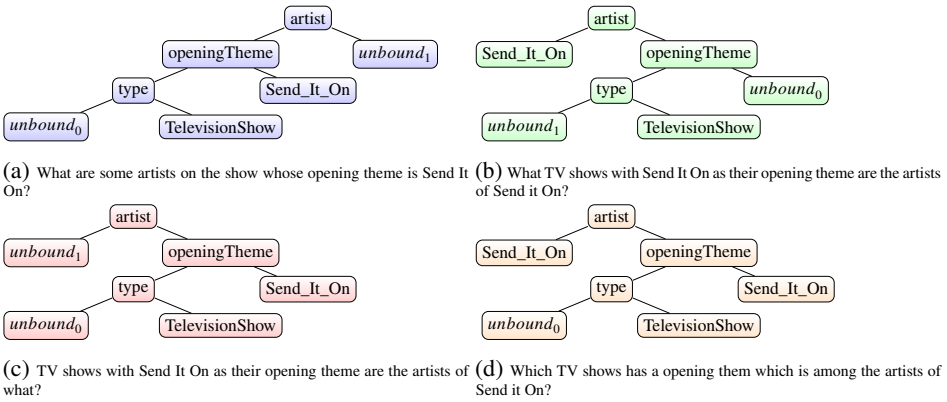
Ranking Model

The ranking model consists of two Tree-LSTMs which are used to map the input walks and question into a latent vectorized representation. We will later apply a similarity function to rank the candidate queries based on the similarity score. In the preparation phase for the input question to the *Question Tree-LSTM*, we substitute the surface mentions of the entities in the question with a placeholder. Then the dependency parse tree is created (see Figure 6.5). Furthermore, the *Query Tree-LSTM* receives the tree representation of the candidate walks. Figure 6.6 depicts the candidate walks of the running example. While all of the candidate walks are valid, only 6.6(a) provides the correct interpretation of the input question.

6.4 Empirical Study

In this section, we discuss the benchmark dataset and compare the result of SQG with two baseline systems. We further provide an extensive analysis of SQG on its different sub-modules such as an evaluation of question type classification, query generation, and ranking model.

SQG is implemented using Python/pytorch. Additionally, we use Glove word embedding [102] as the word representation in the ranking model. The source code of SQG is published on our GitHub



(a) What are some artists on the show whose opening theme is Send It On?

(b) What TV shows with Send It On as their opening theme are the artists of Send it On?

(c) TV shows with Send It On as their opening theme are the artists of what?

(d) Which TV shows has a opening them which is among the artists of Send it On?

Figure 6.6: Tree representation of the candidate walks along with their NL meaning

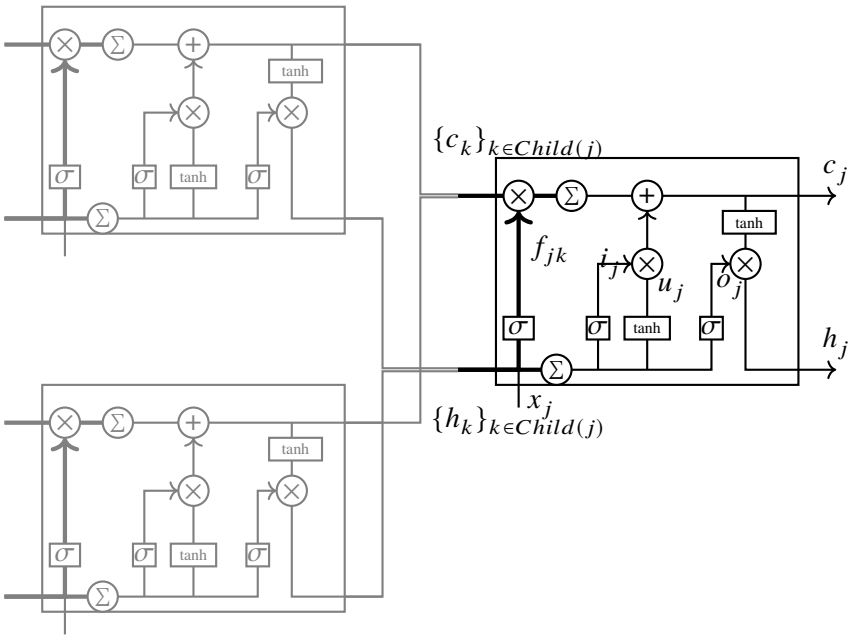


Figure 6.7: The architecture of Tree-LSTM

Table 6.1: The comparison of SQG with existing works

Approach	Precision	Recall	F1-measure
Sina*	0.23	0.25	0.24
NLIWOD*	0.48	0.49	0.48
SQG	0.76	0.74	0.75

* Result of the baseline systems are taken from [9]

page https://github.com/AskNowQA/SQG.

6.4.1 Datasets

We use the *LC-QuAD* [95] dataset in our experiments. The dataset consists of 5,000 question-answer pairs that cover different complexity and types of questions, such as simple and compound, boolean and count. Each pair is also annotated with the corresponding SPARQL query, target entities, and relations. The annotations are compatible with DBpedia [111] (version 2016-04).

6.4.2 Performance Evaluation

Some QA systems use a pipeline approach, but are not modular, hence it would require a major engineering effort to extract their query building components in order to compare against them. However, Singh et al. [9] reported the performance of Sina QB and NLIWOD QB in terms of precision, recall and F1-measure on a subset of LC-QuAD containing 3,200 questions. We compare the performance of SQG on the same subset of LC-QuAD with the same metrics. Table 6.1 demonstrates that SQG significantly outperforms the baseline systems.

There are three shortcomings in the baseline systems. First, they require the correct entity/relation as input as NLIWOD does not support multiple candidates and Sina fails when there are more than three candidates. Second, the query augmentation ability is limited in both systems, and third, they lack a ranking mechanism, which is required to reorder the list of candidate queries with respect to the question. We have addressed these problems in our approach and, in the next section, we first evaluate the feature identification in SQG. Afterwards, we define different scenarios, in which the input of the query generator varies from only target entity/relations to the list of candidates per utterance. Finally, we examine the performance of Tree-LSTM as the ranking mechanism.

Feature Identification

First, we assess the accuracy of two different classifiers. Note that the results are independent of the entity/relation linking module, as they take no additional input from it. We perform a 10-fold cross validation on 50% of the dataset to train the model and find the optimal parameter values. We then evaluate the classifiers using the optimal parameter values on the test set (See Table 6.3). A large amount of diverse training data ensures that the models, albeit simple, perform satisfactorily. Avoiding a manually crafted set of patterns enables SQG to be more applicable in different settings without further manual intervention.

Table 6.3: The accuracy of feature identification on the LC-QuAD Dataset

	Model	Precision	Recall	F1-measure
Question Type	Naive Bayes	0.92	0.92	0.91
	SVM	0.99	0.99	0.99
Duplicate Relation	Naive Bayes	0.87	0.90	0.88
	SVM	0.90	0.91	0.89

Query Generator Evaluation

In order to provide the input for the query generator, we uses two set of tools: TagMe [79] for entity linking along with RNLIWOD [1] for relation linking; and EARL [70] a joint entity/relation linking tool. The performance of EARL was superior on the evaluation dataset and for brevity we report the results using EARL. Below we introduce three scenarios to evaluate the query generator as well as the ranking model in various settings:

Top-1 correct We supply only the correct target linked entity/relation to provide an upper-bound estimation of our performance.

Top-5 EARL+correct We consider a more realistic setting where a list of 5-candidates per entity/relation from EARL [70] is provided. In order to evaluate SQG independent of the linker system, we insert the correct target linked entity/relation if it does not already exist on the list. The purpose of this scenario is to assess the robustness of SQG when the input annotations include several incorrect links as well as the correct ones.

Top-5 EARL We use the output of EARL [70] to evaluate the QG component in a fully functional QA system.

The evaluation results of query generation for all three scenarios are summarized in Table 6.4. In *Top-1 correct*, the query generator failed once to generate any candidate query, while in six other cases none of the candidate query(s) returns the desired answer. In this scenario, the ratio of generated queries per question is close to one, because only the true target entities/relations were given to the query generator. Consequently, the number of valid walks is very low.

In the second row of Table 6.4, which corresponds to *Top-5 EARL+correct*, we observe that the query generator is able to process the noisy inputs and cover 98% of the questions. Furthermore, the average number of generated query per question has increased to 2.25. In the next section, we study how this increase in the average number of generated query affects the ranking model.

The performance in *Top-5 EARL* has dropped dramatically in comparison to the first two scenarios. The main reason is that for 85% of the questions, EARL provided partially correct annotations, which means that either there are utterance(s) in the question which are annotated with an incorrect set of entities/relations, or there are utterance(s) in the question which should not be annotated, while EARL incorrectly annotates them with a set of entities/relations. However, if we consider only the questions where EARL manages to include all the correct target links to SQG, the coverage is again 98%, which is consistent with the result of the artificial injection experiment from *Top-5 EARL+correct*.

[1] https://github.com/dice-group/NLIWOD

Table 6.4: Evaluation metrics of Query Generator

Scenario	Incorrect(%)	No walk (%)	Covered (%)	Avg. #query
Top-1 correct	0.01	0.0	0.99	1.18
Top-5 EARL+correct	0.02	0.0	0.98	2.25
Top-5 EARL	0.12	0.72	0.16	3.28

Table 6.5: The distribution of incorrect and correct data per scenario

Scenario	Size	Train		Dev		Test	
		Correct	Incorrect	Correct	Incorrect	Correct	Incorrect
Top-1 correct	5,930	0.85	0.15	0.84	0.16	0.87	0.13
Top-5 EARL+correct	11,257	0.46	0.54	0.46	0.53	0.48	0.52
Top-5 EARL	4,519	0.20	0.80	0.19	0.81	0.22	0.78

Ranking Model Evaluation

Before we analyze the ranking model, we examine the dataset that is prepared for it. We employ the Stanford Parser [123] to generate the dependency parse tree of the input questions and the generated candidate queries from the output of the query generator to create the dataset for the ranking model. We split the dataset into 70%/20%/10% for the train set, development set, and test set, respectively. Table 6.5 provides some insights on the dataset, on which the ranking model is evaluated.

In the *Top-1 correct* scenario, the number of generated queries per question is 1.18, because there are not many possible ways to connect the linked entity/relations. Moreover, the first row of Table 6.5 demonstrates that the number of incorrect items in the dataset is not proportional to the number of correct items. As a result, the ranking model is given an unbalanced dataset, since the dataset mostly contains positive examples. In *Top-5 EARL+correct*, the number of generated queries is higher, which leads to not only more training data for the ranking model but also more diverse data such that we would not face the same problem as in the previous scenario. The second row of Table 6.5 shows that the distribution of incorrect and correct data is almost equal, which leads to an increase in the performance of the ranking model, as it provides the model enough data on both classes to learn the set of parameters that perform well. In *Top-5 EARL*, the distribution of correct and incorrect instances in the datasets is not balanced, though the average number of generated queries is higher than in the other scenarios. Note that in this scenario, the total number of generated queries is far lower than in the other scenarios, because the 72% of cases no walk is generated. The peak in the number of incorrect instances is caused by wrong annotations, which were provided by the NED and RE component.

We have used two similarity functions in the ranking model: Cosine similarity and a neural network-based function analogous the approach presented in [120]. The neural network computes the distance and the angle between the two latent representations of the query. However, the neural network-based approach had superior performance compared to the cosine similarity function. Thus, in the following, we provide the result of the ranking model using the neural network as the similarity function. Note that for every scenario, the parameters of the similarity network was tuned on the development set. In the interest of comparing the Tree-LSTM ranking model to a simpler model such

Table 6.6: Micro accuracy of the ranking model using the top ranked item

Scenario	LSTM	Tree-LSTM		
	F1 (%)	Precision (%)	Recall (%)	F1 (%)
Top-1 correct	0.54	0.77	0.74	0.75
Top-5 EARL+correct	0.41	0.84	0.84	0.84
Top-5 EARL	0.32	0.73	0.75	0.74

as LSTM, Deep LSTM siamese network for text similarity [124] using word-level embedding and the same similarity function. Table 6.6 demonstrates the superiorty of Tree-LSTM over LSTM in all three scenarios.

In the *Top-1 correct*, despite the unbalanced distribution of the dataset, the ranking model achieves an F1 measure of 74%. This, however, does not accurately reflect the performance results for the ranking model as the average number of generated queries in this scenario is 1.18, which means that mostly there is only one candidate query per question.

As the second row of the Table 6.6 reveals, the F1-measure increases from 75% in *Top-1 correct* to 84% in *Top-5 EARL+correct*. In contrast to the Top-1 correct, the average number of generated queries in *Top-5 EARL+correct* is almost double to 2.25 and the distribution of correct/incorrect example is almost balanced. This results show that the model performs better in comparison to *Top-1 correct* when it is given a bigger and balanced dataset.

A micro F1-measure of 74% was achieved the *Top-5 EARL*. Further analysis reveals that there are two main factors for the drop in the performance: first, the input dataset is highly unbalanced towards incorrect instances; Additionally, the size of the is considerably smaller than the one in *Top-5 EARL+correct*.

6.5 Conclusions

We discussed the challenges of query generation in QA systems and introduced SQG, a two-step SPARQL query generator as a reusable component that can be easily integrated into QA pipelines. In the first step, a set of candidate queries are generated, which is followed by a ranking step that arranges the candidate queries based on their structural similarity with respect to the dependency parse tree of the input question. We provided a detailed analysis of the effect of different factors on the performance of the QG component such as question type detection, noisy input and ranking the candidate queries. In our experiments, SQG outperformed current query generation approaches.

In the next chapter, we aim to extend SQG in order to support more variety of questions that are less represented in question answering datasets, for instance, ordinal and constraint based questions. We show that the modular architecture of SQG enables us to provide a systematic way to support more features, given enough training data.

CHAPTER 7

Query Augmentation

In the last chapter, we introduced SQG, a semantic query builder that supports simple and compound questions as well as boolean and count questions. Through extensive experiments, we showcased its performance perks in comparison with the baseline systems. However, there are a variety of other features that can appear in a question.

In this chapter, we focus on expanding SQG in a systematic way to support new features such as ordinal and filter based questions. In the following section, we discuss the main type of queries along with example questions. Section 7.2 briefly discusses the related works on various techniques, which have been employed to support complex features such as *Ordinal* and *Filter*. We then introduce the overall architecture as well as the details of the approach in Section 7.3 and present the evaluation results in Section 7.4. Section 7.5 concludes our findings.

7.1 Introduction

Most of Question/Answering (Q/A) datasets contain mostly questions with a simple corresponding formal query. Therefore, most of the existing approaches fail to correctly comprehend the challenging questions, in which the query generation task is more demanding. The performance of the query generator depends on the complexity of the input question and the distinct features from the underlying formal query language, which should be supported by the query generator. Nevertheless, given the fact that it is burdensome to define a concrete metric to measure the level of complexity of a natural language question, we establish the complexity of a question based on two features from its corresponding formal query: Type of the formal query (enumerated in Table 7.1) and the number of linked items used in formal query, where the queries that use more linked items, correspond to more complex questions.

In the simplest case, a question can be answered using one entity and one relation from the underlying knowledge graph. In this case, the number of candidate formal queries are limited. For instance, consider the question `Who are the children of Barak Obama`, where the only entity is `Barak Obama` and the relation is `children`. In this example, there are just two possible formal queries that can be built: `SELECT ?c WHERE{?c dbo:Children dbr:Barak_Obama}` and `SELECT ?c WHERE{dbr:Barak_Obama dbp:children ?c}` where the first one is the latter interpretation of the question. However, as the number of linked items increases, the search space might explode and it would be challenging to explore the search space in order to find the candidate queries.

Table 7.1: Various types of Queries and their corresponding sample question

Type	Description/Example/SPARQL
List	The question is a factoid question (single or multiple relations) Example: Who are the children of Barack Obama? SELECT ?child where { dbr:Barack_Obama dbp:children ?child}
Boolean	The question is a yes or no question Example: Is Paris the capital of France? ASK WHERE { dbr:France dbo:capital dbr:Paris ; rdf:type dbo:Place }
Count	The intention of the question is to count the number of the possible results Example: How many cities are in Germany? SELECT COUNT(?city) WHERE {?city dbo:country dbr:Germany ; rdf:type dbo:City}
Ordinal	The question requires ordering of the results over a certain criteria Example: What is the most populated city in Italy? SELECT ?city WHERE{?city dbo:country dbr:Italy ; dbo:populationTotal ?population ; rdf:type dbo:City } ORDER BY DESC(?population) LIMIT1
Filter	The question requires the results to be restricted using a certain criteria Example: List all cities with more than a million population in Egypt? SELECT ?city WHERE{?city dbo:country dbr:Egypt ; dbo:populationTotal ?p ; rdf:type dbo:City. FILTER (?p > 1000000)}

SimpleQuestions [51] and WebQuestions [52] are the de facto standard Q/A datasets based on Freebase [17] as they are used in many of QA over Freebase systems [118, 125–127]. All the questions in SimpleQuestions and more than 80% of WebQuestions can be answered using a single relation in the underlying knowledge graph. Furthermore, there are only 3% *Ordinal* questions and no *Boolean* question in WebQuestions. Consequently, most of the introduced approaches mainly focus on the first type (see *List* in Table 7.1). However, LC-QuAD dataset filled the gap to some extent by providing 7% *Boolean* and 12% *Count* questions out of a total 5,000 questions. As a result, more researchers concentrate on these two types as well [110, 128, 129]. Yet, very limited effort has been spent on the last two categories, in spite of the fact that the number of *Ordinal*/*Filter* questions is increasing in QA datasets. Given that there are already advanced approaches to deal with the first three groups, we aim to enhance an existing query generator in order to not reinvent the wheel and benefit from the existing infrastructures. Among others, SQG [110] reports significantly better accuracy in comparison to other query generator components on various datasets. Thus, in this work we concentrate on extending SQG to support more complex queries, namely *Ordinal* and *Filter*.

7.2 Related Work

The main-stream question answering systems over knowledge graphs can be divided into two categories: Semantic parsing methods and End-to-end approaches. 62 semantic parsing question answering systems from 2010 till 2015 are analyzed Hoffner et al. [14]. They discuss the main challenges in question answering systems as well as the common solutions. Chakraborty et al. [56] provide a more recent overview of neural networks based question answering.

Although end-to-end QA system achieved state-of-the-art results, they mostly focus on simple/-compound question, and either neglect other types [56] or use simple pattern matching techniques to address *Ordinal* or *Filter* types. Hence, we mostly study semantic parsing methods.

Walter et al. [130] introduce BELA - a QA system that consists of a 5-step pipeline: question parsing, template generation, string similarity computation, synonym-finding, and semantic similarity computation. The main idea of the system is that the system decides, which steps should be applied, depending on the complexity of the input question. The system is evaluated on QALD-2[1], however it is not capable of answering questions that require sorting or filtering of the results.

Unger et al. [49] propose TBSL, a QA system that parses the input question to extract syntactic information from the question using predefined lexicons, then it uses this information to generate a logical expression similar to the question. Using this expression, the system chooses the candidate query templates. Finally, TBNSL attempts to fill in the empty slots in the candidate templates through the entities and relations mentioned in the given question. Moreover, it uses ranking techniques to select the best candidate query. Considering that the query templates are manually created based on the dataset at hand, it is considered to be over-fitted for the dataset.

CASIA [131] and SINA [115] are two more examples of QA systems based on a pipeline architecture. The pipeline of such systems includes tasks such as question processing, entity and relation recognition and disambiguation, and SPARQL query generation. These systems are benchmarked on the QALD-3[2] challenge dataset. Similarly, there is no support for questions where complicated features such as *Filter*, *Ordinal*, etc. are required.

Hakimov et al. [132] develop a QA system that uses a semantic approach based on Zettlemoyer et al. [133]. They investigate the use of handcrafted lexicons to minimize the lexical gap between the vocabulary used in natural language questions and the one of the training data. The system is benchmarked on QALD-4[3], however, the systems has no support for *Ordinal* or *Filter* questions.

POMELO [134] is another pipelined QA system, which resembles the architecture of CASIA [131] and SINA [115]. In addition to the pre-processing steps in the pipeline, POMELO scans the question for certain terms such as numbers, `mean`, `higher`, etc. in order to construct the SPARQL query. This step helps POMELO to support more query types than CASIA [131] and SINA [115]. However, it fails to handle compound questions. Moreover, its support for *Filter* and *Ordinal* types is limited due to the fact that it is based on a hand-crafted list of patterns.

The AskNow approach as described in [48] is a QA framework by M. Dubey et al. that takes a natural language question as its input, then transforms the question into an intermediate logical form called *Normalized Query Structure*, which later will be changed into a SPARQL query. AskNow defines three types of queries: Simple, complex and compound. As a result, it is able to support compound

[1] http://qald.aksw.org/index.php?x=task1&q=2
[2] http://qald.aksw.org/index.php?x=task1&q=3
[3] http://qald.aksw.org/index.php?x=task1&q=4

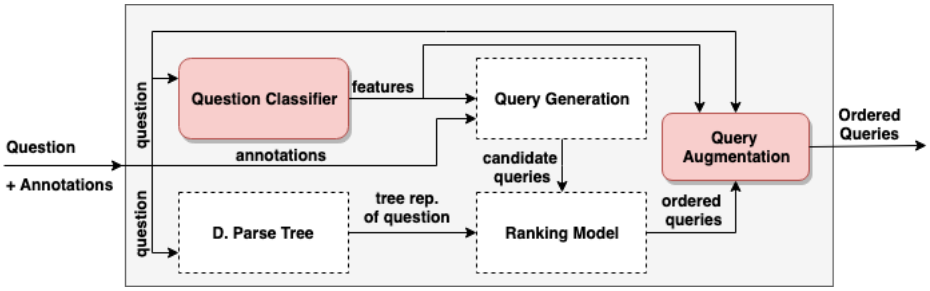


Figure 7.1: Proposed ExSQG Architecture. Components highlighted in red are modified/added components

questions. Nevertheless, the support for *Filter* and *Ordinal* is limited to the pre-defined patterns. Much like the system proposed by Unger et al. [49], Abujabal et al. [113] use a similar approach with the main difference that the system is able to learn SPARQL templates from question-query pairs. Given a question, it tries to match the question to an empty candidate template(s) that corresponds to the given question. In addition, it benefits from ranking methodologies for selecting the best candidate queries.

The aforementioned QA systems are either based on templates/patterns or use and ad-hoc methods to support complex queries. While, we base our solution on extending a well structured, modular, standalone SPARQL query generator that is capable of generating target SPARQL queries for input questions, provided the entities and relations mentioned in the question.

7.3 Approach

Given a question in natural language and the correct linked items (entities and relations), SQG [110] goes into the details of generating a SPARQL query that corresponds to the input question. By using this generated SPARQL query and augmenting it with necessary constraints, we are able to obtain a SPARQL query that supports new, previously unsupported, types.

In order to extend SQG [110] to support the two new types, we model these types as extra constraints that need to be applied on the list of all possible answers. For the *Ordinal* class, to get the correct answer for the example question **Q1:**`"What is the most populated city in Italy?"`, we first need to get a list of all the cities in Italy, then sort them in descending order with respect to the population of each city and then return the top city as the most populated city in Italy. The same idea applies to the type *Filter*, where the list of possible answers should conform to a certain constraint. For example, given the question **Q2:**`"What are the cities with more than a million population in Egypt?"`, we need to get all the cities in Egypt and only return those with the population more than a million as the answer. This unified view of modeling the new types as constraints enables us to extend SQG by adding an extra layer over the existing architecture.

To support the aforementioned types, we divide the overall task into three sub-tasks. First, we need to classify the given questions in order to recognize those questions that belong to the new types. Second, we parse the given question to extract special keywords that would help us to select a KG property, which would act as the constraint for the intended SPARQL query. The last task is to set any parameters needed for the SPARQL query in order to capture the intention of the given question.

Figure 7.1 shows the architecture of ExSQG. It extends SQG [110] with two new components – *Question Classifier* and *Query Augmentation*. The new question classifier replaces the original

question classifier from the SQG [110] as it does not support the new question types. The original question classifier is built as a flat classifier using Naive Bayes and SVM and supports only List, Boolean and Count questions.

In SQG [110], the ranking model was the last step in the query generation pipeline. However, in the ExSQG architecture, the query augmentation component resides at the end of the pipeline. The augmentation component is responsible for complementing the SPARQL query, which is selected by the ranking model, by adding the necessary constraints and parameters in order to generate the final query that corresponds to the input question.

Intuitively each question is of List, Boolean or Count type. However it may belong either to *Ordinal* or *Filter*, or both. We call the first three categories PRIMARY CLASSES and *Ordinal* and *Filter* secondary classes. Accordingly, we build a hierarchical question classifier, which consists of a multi-class classifier for primary classes and a binary classifier for each of the secondary classes. Figure 7.2 shows the architecture of the *Questions Classifier*. When a question is passed through the classifier module, it is first classified by the primary classifier to find out its primary class. Given the primary class, it passes through all the secondary binary classifiers to check if the question belongs to one or more of the secondary classes. As shown in Figure 7.3, both **Q1** and **Q2** are identified as *List* by the primary classifier, however, **Q1** is classified as *Ordinal* as the secondary class, while the second class of **Q2** is established as *Filter*.

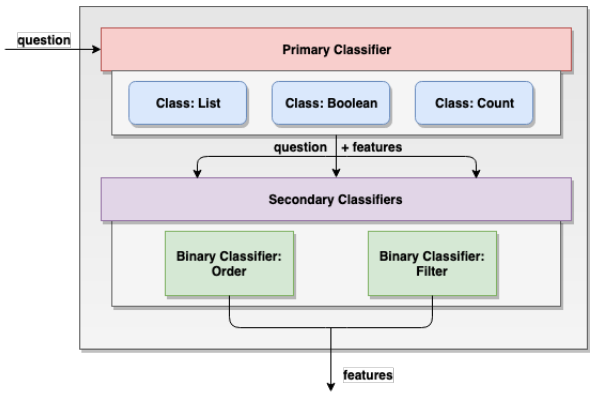


Figure 7.2: Architecture of the Hierarchical Question Classifier

After the question is classified, it passes through the rest of the pipeline. If the question is classified to have only a primary class and no further secondary classes, then the query is returned by the ranking model as the result of SQG [110]. On the other hand, when the question is classified to be one of the secondary classes, it passes through the query augmentation component with its corresponding SPARQL query chosen by the ranking model.

The first task of the query augmentation is to select a KG property that acts as the constraint in the SPARQL query. First, the natural language question is cleaned by removing stop words and any entity mentions. The result of this process is called a *base-form* and is used in the *Parameters Settings* step. By parsing the base-form according to the class of the question provided by the question classifier, we are able to further clean it, which would result in having single or multiple words. This sequence of words is called *keyword* or *keywords*. For example, the base-form for the **Q1** is "`most populated city`" and the keywords are "`most populated`".

In parallel with the keyword extraction task, the SPARQL query provided by the ranking model is

used to capture the list of KG relations in the one-hop distance of the subgraph containing the answer. Empirically, by analyzing Filter and Order questions and their corresponding gold SPARQL queries. We found that the relations used as constraints are always in the one-hop space distance from the subgraph that contains the answer. Thus, we operate under the assumption that the KG property that acts as the constraint is contained within this list. These extracted relations are then filtered retaining only those, which are comparable (e.g. Numbers, Dates, etc.). For instance, the candidate relations for **Q1** are `dbo:areaTotal, dbo:Country, dbo:populationTotal`, etc. .

In order to select the correct KG relation from the list of possible relations, we capture the semantic closeness of the keywords and each of those relations by computing the cosine similarity between their word embeddings. The KG relation and keywords, which form the closest pair, are selected as the final KG relation, which acts as the constraint in the final SPARQL query. Note that since both the keywords and KG relations might consist of more than a single word, we use Word Mover Distance [108] to measure the similarity between the keywords and the KG relations. For example, from the list of candidate relations for **Q1**, `dbo:populationTotal` is the most similar one in comparison to the keywords `most populated`. It's worth noting that before checking the similarity between the KG relations and the keywords. The KG relation is transformed into a correct English form, from `populationTotal` into `population total`. This is done by simply splitting the KG relation at each capital letter, since they are always written in a camel case form. The final step is to set any parameters for the given query. This parameter setting depends on the type of the query. For queries of type *Ordinal*, there are three parameters to be considered; the direction of sort, offset and limit. In order to set the direction of sort, we train a classifier that predicts the sorting direction given the keywords. On the other hand, the offset is set by parsing the base-form provided by the components responsible for the keyword extraction. If the base-form contains an ordinal mention (e.g first, second, third, etc.), it is used to set the offset in the SPARQL query. Otherwise, the offset is set to zero. The last parameter in the *Ordinal* queries is the limit. To set the limit of the query, we use Part Of Speech(POS) tags to check if the keyword or keywords refer to a singular or plural noun to set the limit accordingly. For our running example question **Q1**, the limit would be set to one as the keywords `most populated` refers to `city`, which is singular. Otherwise, it is set to negative one, which means all possible answers.

If the query belongs to the *Filter* class, there is only one parameter to be set, which is the comparison operator (e.g. less than, more than, same as, etc.). In order to be able to set the correct operator, we train a classifier that predicts the operator given the keyword. The keywords are prepared by running the keyword identification component on the training sets. The classifier is trained on such keywords and their corresponding operator extracted from each SPARQL query. For instance, the operator *greater* with operand `1000000` would be extracted for the example question **Q2**.

Finally, after the KG property is selected and the values of the parameters are set, these results are used to augment the SPARQL query provided by the ranking model. This augmentation is done as follows i) first, we syntactically parse the query returned from the ranking model ; ii) we prepare the SPARQL equivalent for any of the parameters and/or constraints; iii) we append these additions to the query returned by the ranking model.

Figure 7.3 illustrates the flow of ExSQG with the example questions **Q1** and **Q2**. It shows each component in the pipeline with its inputs and its output when the system is given an *Ordinal* or *Filter* question.

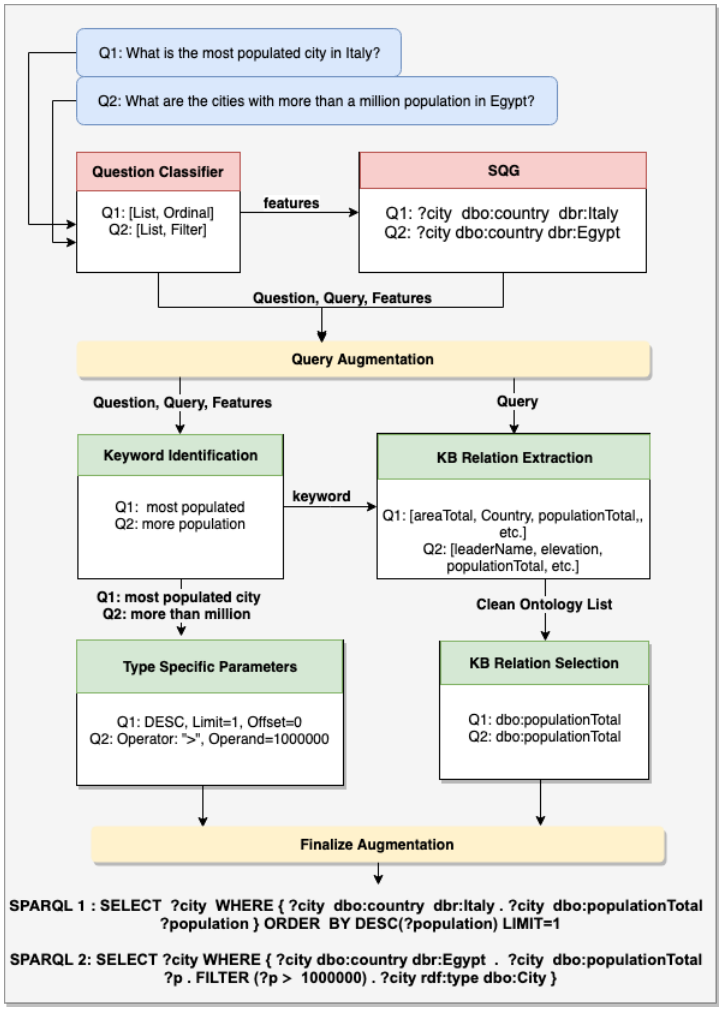


Figure 7.3: ExSQG pipeline for Ordinal and Filter examples

7.4 Empirical Study

In this section, we introduce the datasets used in this work and provide statistical information about them. In addition, we present the results of ExSQG on the benchmarking datasets.

7.4.1 Datasets

Q/A datasets commonly contain triples of i) natural language question, ii) the equivalent formal query, and iii) the answer set. Since many of the Q/A datasets only contain *List* questions with no extra features such as *Ordinal* and *Filter*, we hand-picked the ones that include such questions from multiple datasets, so that we could build a robust and general query generator. In order to have a unified dataset, we aim to collect the datasets with the same underlying KG. Among others, DBpedia [111] is an ongoing community-based knowledge base that is in a constant process of development and we would use the datasets, which are based on DBpedia.

First, we use LC-QuAD[54], which contains 5,000 manually crafted questions and their corresponding SPARQL query. Although LC-QuAD does not contain any questions that belong to the new types, we include it in order to provide performance comparison with the baseline system (SQG). Second, we use all the datasets from the QALD [4] challenge (QALD 1-9). As these datasets where part of a Q/A campaign over multiple years, many of the questions are used more than once in these datasets (out of more than 5,000 questions in these datasets, only 1400 are unique). However, these datasets are particularly important since they contain all of the types of questions, and they are carefully designed to challenge different aspect of the QA systems. The last dataset we use is DBNQA [135], which is a template-based dataset containing about 800,000 automatically created question and SPARQL query pairs. This dataset is especially useful, since it provides a vast number of questions from the *Filter* and *Ordinal* types.

Although DBNQA contains the target question types, it is generated using a set of pre-defined templates. Thus, if we train the classifiers on DBNQA, it would be biased towards the underlying templates. On the other hand, considering the number of unique question/query pairs in the QALD 1-9 challenge, it is not sufficient to train the classifiers. Therefore, we combine training and testing sets from all the available datasets.

The idea behind these combinations is to compare the performance of the models trained on each of the combinations with each other. These combinations are as follows:

- LC-QuAD: Using only LC-QuAD
- LC-QuAD + QALD: Combined data from both datasets
- LC-Quad + QALD + DBNQA: Combined data from all the datasets

Since DBNQA has over 800,000 questions-query pairs, while LC-QuAD and QALD contain about 10,000 questions combined, we do not include DBNQA entirely but rather use a subset of the dataset in order to avoid the classifiers' overfitting over questions from DBNQA. We used random different subsets with different sizes that varied between **1%**, **5%**, **10%**, and **25%** from the available questions in the dataset.

Using these multiple subsets gives us a better idea when the model gives the best performance, while decreasing the chance of overfitting over DBNQA. The combined datasets are named as follows:

- LC-QuAD + QALD + 1% DBNQA: Combined 1
- LC-QuAD + QALD + 5% DBNQA: Combined 5
- LC-QuAD + QALD + 10% DBNQA: Combined 10
- LC-QuAD + QALD + 25% DBNQA: Combined 25

For the secondary classifiers, we prepare the training and testing sets using all the available data from all the available datasets. Since the amount of the available data for the secondary classes is not as much as the data available for the primary classes.

Table 7.2 shows the total number of question and query pairs per dataset. In addition, it shows the total number of questions available for each type per dataset.

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

Table 7.2: Datasets Statistics

Dataset	# of Questions	Unique Questions	List	Boolean	Count	Ordinal	Filter
QALD (1-9)	5,237	1,396	1,056	98	79	94	75
LC-QuAD	5,000	4,998	3,967	368	658	0	0
DBNQA	894,499	871,166	688,689	76,835	98,372	3,893	1,797

7.4.2 Experiment Settings

For the training process for any of the aforementioned classifiers, we prepared a train/test set from all the available data. We split each dataset as 70% for the training set and 30% for the test set. Furthermore, we use 10-fold cross-validation during the evaluation process. In addition, we use *scikit-learn*[5] implementations for all the classifiers used.

Moreover, for the cleaning process of questions, we use Spacey[6] and NLTK[7]. Finally, to prepare the embedding matrix, which contains the vector representation for all the words in our vocabulary, we use the pre-trained word vectors by Global Vectors for Word Representation (GloVe)[8] [102].

7.4.3 Evaluation Metrics

Since the proposed system architecture consists of a pipeline of components, in order to evaluate the performance of such a system, we first evaluate the performance of each component individually. Then we assess the overall performance of the system.

We evaluate the performance of the classifiers trained in terms of *accuracy*. In addition, we use *precision*, *recall*, and *F1-score* to measure the performance of the KG property selection component.

7.4.4 Empirical Results

The selection process of the best classifier consists of two parts. First, we select the best classifier with the best set of features. Then, we experiment with the best performing classifier with the best set of features against different datasets with various sizes.

Table 7.3 shows the accuracy of the question classifier under a different set of features. This experiment is done on the **combined dataset 5**. In order to select the best set of features, we show the accuracy of the classifier at each row for the current feature, combined with the best set of features selected so far. As the table shows, we end up using the *MaxEnt* classifier as it out-performed the other classifiers.

Table 7.4 shows the performance of the classifier when it is trained on different datasets. In this experiment, we use the **combined dataset 25** as the test for all the classifiers. We can see from the table that the performance of the question classifier increases with the size of the dataset. However, this increase could also be due to the classifiers overfitting over questions from DBNQA.

[5] https://scikit-learn.org/stable/
[6] https://spacy.io/
[7] https://www.nltk.org/
[8] https://nlp.stanford.edu/projects/glove/

Table 7.3: Accuracy for the question classifier under different features

Feature	NB	SVM	MaxEnt
1-gram	91.0%	96.7%	98.5%
(1+2)-grams	95.3%	96.9%	98.9%
(1+2+3)-grams	95.7%	96.7%	98.9%
+TF-IDF	94.5%	92.4%	99.0%
+Normalized Numbers	95.7%	96.9%	99.0%
+POS	95.9%	96.4%	99.1%
First N-words N=3	93.6%	94.2%	96.2%
First Last N-words N=3	93.3%	95.3%	97.4%

Table 7.4: MaxEnt Classifier Performance against multiple datasets of different sizes

Dataset	MaxEnt
LC-QuAD	90.1%
LC-QuAD + QALD	89.7%
Combined_1	95.9%
Combined_5	99.3%
Combined_10	99.5%

For the following experiments, we mainly focus on the QALD datasets to show the performance of the system as they are very popular and used a lot in benchmarking QA over KG systems [10]. Thus, we are able to have a reference point to compare our approach with other systems.

Table 7.5: Accuracy of the question classifier on QALD (4, 5, 6, 7)

Dataset	No. Questions	Accuracy
QALD-4	67	51 (76%)
QALD-5	33	28 (84%)
QALD-6	99	87 (87%)
QALD-7	30	25 (83%)

Table 7.5 shows the accuracy of the hierarchical question classifier on QALD (4, 5, 6, 7). It also shows the total number of questions available per dataset. The accuracy of the proposed question classifier in Table 7.5 is less than the reported accuracy for the question classifier for SQG [110], because of the complex nature of the questions that belong to secondary classes.

Table 7.6 shows the precision, recall, and F1 score for ExSQG for questions of type *Ordinal*. A generated SPARQL query is considered correct if it yields the same answer as the target SPARQL query, this means that the system is able to correctly classify the question and successfully generate the correct SPARQL query. The performance of the ExSQG is lower than the performance on QALD-5, and 6 for two reasons. First, by inspecting the questions that lead to an incorrect answer, we found out that the number of miss-classified questions from QALD-4 is higher than those of QALD-5, and 6.

Table 7.6: Performance of Ordinal Questions Pipeline

Dataset	Precision	Recall	F1
QALD-4	0.40	0.33	0.36
QALD-5	0.83	0.83	0.66
QALD-6	0.80	0.66	0.72
QALD-7	0.33	0.50	0.40

Second, most of the questions that belong to the *Ordinal* class from QALD-4 were generally more complex than those that belonged to QALD-6. Not in terms of linked items, rather in the queries that correspond to the question and the constraints used in such queries. For example, some query constraints are not simply KG relations but a count over such relations.

Table 7.7: Performance of Filter Questions Pipeline

Dataset	Precision	Recall	F1
QALD-4	0.11	1.00	0.20
QALD-6	0.14	0.33	0.20

Table 7.7 shows the precision, recall, and F1 score of ExSQG for questions of type *Filter*. It also shows that the ExSQG system does not provide the same performance as it does for questions of the type *Ordinal*. This is due to the fact that there are much fewer questions of the type *Filter* that we support in the datasets than questions of type *Ordinal*. The current system is able to correctly generate the SPARQL for questions that require filtration over the value of a KB Relation (e.g. *"Cities in Germany with area larger than 30000 KM"*), or questions that compare two KB resources over a certain KB relation (e.g. *"Does Game of Thrones have more episodes than Breaking Bad"*). In the first question the constraint is the `dbo:areaTotal` and in the second one – `dbo:numberOfEpisodes`. On the other hand, questions that require a string matching filter query, date matching, or filtration based on a count are not yet supported. Therefore, any miss-classification or incorrect query generation would significantly impact the overall performance. The results for QALD-5,7 are not shown as well in this table, because there were only 3 filter questions and our system was not successful to correctly predict and answer them.

Table 7.8: Absolute increase percentage in performance between the SQG [110] and ExSQG

Dataset	No. of Questions	Performance Increase
QALD 4	67	8.0%
QALD 5	33	18.0%
QALD 6	99	5.0%
QALD 7	30	3.0%

Table 7.8 shows the absolute difference in performance between SQG [110] and the ExSQG. For this experiment, we assume an ideal scenario for the question classifier for both systems – SQG [110]

and ExSQG. We also assume that we always get an intermediate SPARQL query from the ranking model for questions that belong to the new types. These conditions are assumed to mitigate any error propagation from SQG [110] and to be able to measure the performance of the ExSQG on questions that belong to the new types. The variation of the performance of ExSQG on QALD (4, 5, 6 and7) as shown in Table 7.8 is due to the fact that there is only a limited number of questions that belong to secondary classes in these datasets. However, there are more questions that have secondary classes in QALD-5 in comparison to the other datasets.

7.5 Conclusions

Encouraged by the existing efforts on query generation in the QA community, we presented ExSQG as an extension to an available query generator component (SQG [110]) in order to support *Filter* and *Ordinal* questions. We provided a hierarchical architecture for a question classifier, which yields high accuracy in different benchmarking datasets. Furthermore, ExSQG augments the query using identified keywords from the question and match them to the linked items from the KG based on word embedding techniques. Finally, we empirically showed that ExSQG achieves state-of-the-art accuracy on the benchmarking datasets.

This wraps up our efforts with respect to the **RQ-2** that is to discover and improve low performing components in semantic question answering. In the chapter that follows, we aim to address **QG-3** that is to involve users to guide the question answering system in order to capture the correct intention of the questions while maintaining user satisfaction.

CHAPTER 8

Interactive Question Answering

In the last three chapters, we focused on the **RG2**, to spot and boost the bottleneck components. In Chapter 5, we presented a shallow parser, that is tailored for semantic question answering parsing and its distantly supervised setting. Furthermore, we introduced a semantic query builder with a ranking mechanism in Chapter 6. It was further extended to increase the support for more complex features such as ordinal and filter in Chapter 7.

In this chapter, we discuss our contributions regarding **RQ-3**, to investigate user interactions with semantic question answering to increase the overall performance of the system. We present an interactive framework that enables users to guide the question answering system to find the correct intention of the input question. The next section describes the necessity of an interactive schema, its challenges and the overall idea of our approach. Then, in Section 8.2 we present the user interaction scheme of IQA. Following that we describe the realization of the IQA pipeline in Section 8.3. The evaluation setup is described in Section 8.4. Our evaluation results are presented in Section 8.5. Section 8.6 discusses related work. We provide a conclusion in Section 8.7.

8.1 Introduction

Current semantic QA approaches are capable of effectively answering rather simple factual questions that contain a limited number of entities and relations.

In the case of complex questions, i.e., questions that involve multiple entities and relations, the performance of the existing SQA approaches is still limited. These limitations can, to a large extent, be attributed to the inherent uncertainty associated with the results of the individual pipeline components along with the propagation of errors of the component results through the entire SQA pipeline. This uncertainty often leads to imprecise question interpretations, especially for complex questions.

Figure 8.1 illustrates this problem using an example question from LC-QuAD [136] - a state-of-the-art dataset for evaluation of Semantic Question Answering systems: *"List software that is written in C++ and runs on Mac OS."*. An SQA pipeline incrementally transforms the input question in a semantic query, using components such as a *Shallow Parser* (SP), an *Entity Linker* (EL), a *Relation Linker* (RL) and a *Query Builder* (QB). First, the Shallow Parser identifies keyword phrases "software", "written", "C++", "runs" and "Mac OS". Then the Entity Linker and Relation Linker map these keyword phrases to the entities and relations in the DBpedia knowledge graph. To obtain correct

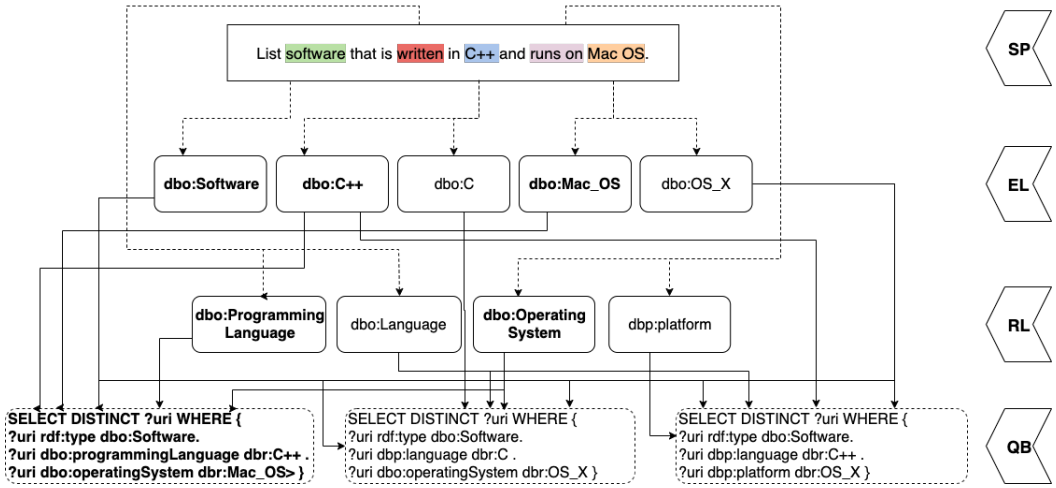


Figure 8.1: An example transformation of a question from the LC-QuAD dataset in possible semantic queries over the DBpedia knowledge graph using an SQA pipeline consisting of a Shallow Parser (SP), an Entity Linker (EL), a Relation Linker (RL) and a Query Builder (QB).

interpretation, the Entity Linker should link the keyword phrase "C++" to the entity *dbr:C++*[1], the programming language and "Mac OS" to the entity *dbr:Mac_OS*[2], the operating system. The Entity Linker should not confuse "C++" with e.g., *dbr:C*[3], another programming language. The Relation Linker should link the keyword phrases "written" to the relation *dbo:programmingLanguage*[4] and "runs on" to the relation *dbo:operatingSystem*[5]. Here, the task of relation linking is particularly difficult due to the lexical gap, the required domain knowledge, and the ambiguity of the candidates. To reduce the number of candidates, the Relation Linker can rely on the Entity Linker results, e.g., by taking into account the relations of the linked entities in the knowledge graph. Finally, the Query Builder component utilizes the results of the Entity Linker and Relation Linker to build the semantic query. Errors in the results of the Entity Linker and Relation Linker can often lead to the misinterpretation of the user question. With an increasing number of entities and relations mentioned in the user question, the likelihood of such errors increases.

The objective of this chapter is to address the limitations of the existing SQA approaches in answering complex questions through the provision of a novel user interaction scheme. While other domains like Information Retrieval and keyword search over structured data take significant advantage of user interaction models (e.g., [137]), such models are not yet widely adopted in the context of Semantic Question Answering. The proposed IQA scheme can be particularly beneficial in answering complex questions when the intended semantic interpretation of the question cannot be accurately inferred using automatic methods. From the algorithmic perspective, this scheme can facilitate SQA systems to reduce uncertainty during the query interpretation process efficiently. From the user perspective, this scheme can empower users in effectively guiding SQA algorithms towards the

[1] http://dbpedia.org/resource/C++
[2] http://dbpedia.org/resource/Mac_OS
[3] http://dbpedia.org/resource/C
[4] http://dbpedia.org/ontology/programmingLanguage
[5] http://dbpedia.org/ontology/operatingSystem

intended results.

Given an SQA pipeline and a user question, the goal of IQA is to facilitate an efficient and intuitive generation of the intended question interpretation through user interaction. The proposed interaction scheme incrementally refines user questions in the intended semantic queries by requesting user feedback on several items called *interaction options*. The main challenge to be addressed here is the trade-off between the efficiency and the usability in the interaction scheme. In this context, efficiency refers to the minimization of the interaction cost (i.e., the number of requests for user feedback). The usability means the ease of use/understandability of the interaction options. To the best of our knowledge, none of the state-of-the-art SQA systems support user interaction in Semantic Question Answering in the way envisioned in this chapter.

Overall, in this chapter we make the following contributions:

- We present a probabilistic foundation to estimate the likelihood of the interaction options based on the formalization introduced in Chapter 4. This model builds a basis for the systematic generation of effective interaction options in a variety of categories.
- We propose a user interaction scheme that seamlessly incorporates user feedback in the Semantic Question Answering process to reduce uncertainty efficiently. We adopt a cost-sensitive decision tree to balance the trade-off between usability and efficiency of the options in the interaction process.
- We incorporate the usability of interaction options into a new metric, *Option Gain*, that balances the usability and efficiency of interaction options and facilitates the selection of interaction options that are efficient and intuitive for the user.
- We showcase an instantiation of the proposed user interaction scheme in a web-based IQA prototype while utilizing existing components developed by the SQA community.

We demonstrate the effectiveness and efficiency of the proposed interaction scheme for Semantic Question Answering in an extensive experimental evaluation and a user study. Our evaluation results on LC-QuAD, an established dataset for the assessment of Semantic Question Answering systems, demonstrate that IQA can significantly improve the effectiveness, efficiency, and usability of Semantic Question Answering systems for complex questions. In particular, the IQA-OG configuration that adopts Option Gain achieves an increase of up to 20 percentage points in terms of F_1 score compared to the baselines on a subset of LC-QuAD utilized in the user study. Furthermore, this configuration enhances the ease of use as reported by the users.

8.2 IQA User Interaction Scheme

Given a user question Q and a large-scale knowledge graph $\mathcal{KG}$, a Semantic Question Answering pipeline PL can generate a large number of possible complete question interpretations. We denote the set of all complete question interpretations of Q generated by PL given $\mathcal{KG}$ as a *question interpretation space* QIS (Notations frequently used are summarized in Table 8.1).

IQA facilitates an efficient and intuitive generation of the intended question interpretation through a user interaction scheme. In IQA, an *interaction option* IO is a unit adapted for user interaction.

Table 8.1: Summary of frequently used notations.

Notation	Description
$Q = (q_{NL}, QN)$	a representation of the user question
q_{NL}	a user question as a natural language expression
QN	a multiset of information nuggets
QI	a partial question interpretation
CQI	a complete question interpretation
plc	an interpretation function
QIS	the question interpretation space
IO	an interaction option
OG	Option Gain
IG	Information Gain

The goal of the interaction scheme is to reduce the question interpretation space QIS with each user interaction efficiently while providing intuitive interaction options.

Conceptually, the IQA interaction scheme resembles the induction of a cost-sensitive decision tree [138], where the cost reflects the complexity and usability of the interaction options from the user perspective. We rely on the notion of Option Gain introduced later in this section to facilitate the usability and efficiency of the interaction scheme.

8.2.1 Interaction Options and Subsumption Relation

An *interaction option* IO is a unit adapted for user interaction to reduce the question interpretation space QIS. In IQA we group interaction options in the following categories: 1) nugget interpretations, 2) superclasses and types of entities, 3) answer types of semantic queries, and 4) complete question interpretations (i.e., semantic queries).

To facilitate an effective reduction of the question interpretation space QIS by interaction, we establish a subsumption relation between interaction options and complete question interpretations.

We say that an interaction option IO subsumes a complete question interpretation $CQI = (QI, AT, QG)$ if one of the following conditions applies:

C1. Interaction option IO represents a nugget interpretation leading to the generation of the semantic query, namely: $IO \in QI$.

C2. Interaction option IO is a superclass or a type of an entity included in CQI: there must be a URI x in the query graph QG of the complete query interpretation CQI, for which a triple

$$(x, rdfs : subClassOf, y),\ or(x, rdf : type, y)$$

exists in the knowledge graph, and $y \equiv IO$.

C3. Interaction option IO represents the answer type of CQI: $IO \equiv AT$.

C4. Interaction option IO is equivalent to the semantic query: $IO \equiv CQI$.

8.2.2 Option Gain

Interaction options vary concerning their complexity and usability. Complex interaction options can be difficult to understand for the users, potentially leading to an error-prone interaction process (i.e.,

wrong user decisions) and decreasing an overall user satisfaction.

The key concept of the IQA interaction scheme is the *Option Gain* $OG(IO)$. Option Gain takes into account the $usability(IO)$ and the efficiency of the interaction option IO expressed using its Information Gain $IG(IO)$. We define the Option Gain as:

$$OG(IO) = usability(IO)^{\omega} \times IG(IO), \tag{8.1}$$

where $\omega \in \mathbb{N}$ is a parameter that controls the bias introduced by the usability of an interaction option IO in the interaction process, such that by $\omega = 0$ the Option Gain corresponds to the Information Gain without the usability bias.

In IQA the usability of an interaction option is reflected through the usability score $usability(IO) \in [0, 1]$, where 1 corresponds to the most intuitive options and 0 to the most complex options:

$$usability(IO) = \frac{1}{1 + complexity(IO)}. \tag{8.2}$$

The complexity of an interaction option $complexity(IO)$ can be characterized through the syntactic similarity of the interaction option to the initial user question, the degree of abstraction, and the structural complexity.

Given the user question Q, the uncertainty of the question interpretation is the result of several factors, including: F1) the ambiguity of information nuggets in Q and the resulting uncertainty when interpreting these nuggets in a large-scale knowledge graph; F2) the uncertainty of the expected answer type; and F3) a variety of possible graph structures connecting nugget interpretations in a semantic query. Interaction options proposed in IQA aim to reduce this uncertainty.

In the following, we discuss the complexity estimation of the interaction options, which were introduced in Section 8.2.1 above.

C1. An interaction option IO in this category is a nugget interpretation. Intuitively, an IO syntactically similar to the nugget in the user question may appear familiar, and thus less complex, to the user. Therefore, we estimate the complexity of an option IO in this category as the dissimilarity between the information nugget corresponding to the IO in the user question and the representation (e.g., a label) of the IO shown to the user in the interaction process. We adopt the Longest Common Substring (LCS) as a string similarity metric, as this metric was shown to be suitable for short phrases [107].

C2. An interaction option in this category is a superclass or a type of an information nugget contained in the semantic query. The usability of such options depends on the degree of abstraction. We assume that less abstract categories such as "person" and "actor" can appear more intuitive to the users than more abstract categories, such as "living thing". To reflect this intuition, we measure the complexity of the interaction options in this category as the length of the shortest path between the IO and the element of the knowledge graph that directly maps to the corresponding information nugget in the user question.

C3. An interaction option in this category represents an answer type of the semantic query. Given a relatively straightforward set of possible answer types, we set $complexity(IO) = 0$ for the

options in this category.

C4. The interaction options in this category are semantic queries. Intuitively, more complex queries that include a high number of nugget interpretations can appear more difficult to understand from the user perspective. Therefore, we compute the complexity of an interaction option in this category as the number of nugget interpretations it includes.

8.2.3 Information Gain

For the computation of the Information Gain of an interaction option in the question interpretation space QIS, we build upon the probabilistic model proposed in our previous work [137]. We summarize the computation of the Information Gain in the following.

Let $H(QIS)$ be the entropy of the probability distribution in the question interpretation space QIS. The Information Gain of an interaction option $IG(IO)$ is computed as the entropy reduction given user feedback on IO.

Let QIS_{IO} be the set of complete question interpretations in QIS subsumed by IO, and $QIS_{\overline{IO}}$ be the set of all other complete question interpretations in QIS. Furthermore, let $P(IO)$ be the probability that the interaction option IO subsumes the user-intended complete question interpretation.

The entropy of the probability distribution in the question interpretation space QIS is computed as:

$$H(QIS) = -\sum_{CQI \in QIS} P(CQI|Q, \mathcal{KG}) \times log_2 P(CQI|Q, \mathcal{KG}). \tag{8.3}$$

Then, Information Gain of the interaction option is computed as the uncertainty reduction provided by this option:

$$IG(IO) = H(QIS) - \left(P(IO) \times H(QIS_{IO}) + P(\overline{IO}) \times H(QIS_{\overline{IO}}) \right). \tag{8.4}$$

The probability of an interaction option $P(IO)$ is computed as the sum of the probabilities of complete question interpretations subsumed by this option:

$$P(IO) = \sum_{CQI \in QIS_{IO}} P(CQI|Q, \mathcal{KG}). \tag{8.5}$$

8.2.4 User Interaction Process

The conceptual process of the interactive question interpretation using a generic semantic Question Answering pipeline presented in Chapter 4 can be modeled as follows:

Step 1 (SQA Pipeline Execution): The user issues the question Q. The SQA pipeline is executed to generate the question interpretation space QIS.

Step 2 (Pre-Processing): The partial and complete question interpretations generated by the pipeline are utilized to generate the interaction options. Then the subsumption relations between these options and the complete question interpretations in QIS are established.

Step 3 (User Interaction): At each step of the interaction process, the user is simultaneously presented with:

- The interaction option IO^* with the highest *Option Gain*, and
- The most likely complete question interpretation CQI^* in the interpretation space QIS (in a natural language and a semantic representation).

For simplicity, we model the interaction process as a list of binary user decisions, i.e., we assume that the user is presented with one interaction option at a time. In practice, this process can be generalized to present several interaction options simultaneously.

At each step of the interaction process, the user has the following means to interact with the system:

- Accept the interaction option IO^*, i.e., confirm that the presented interaction option correctly interprets (a part of) the question Q.
- Reject the interaction option IO^*.
- Accept the complete question interpretation CQI^*, i.e., confirm that this interpretation correctly reflects the intention of the question.

After each interaction, CQIs that do not comply with the user decision are removed from the question interpretation space QIS using subsumption relation. The Option Gain of all the interaction options is recomputed. The interaction process continues with the currently top-scored IO^* and CQI^*.

The interaction process for a question terminates if one of the following applies:

- The user accepts the complete question interpretation CQI^*.
- The question interpretation space QIS is empty, i.e., the correct interpretation cannot be identified given user feedback.
- The user terminates the process.
- The number of interactions or the time spent by the user reached a threshold.

8.3 Realization

In this section, we present the realization of the proposed IQA approach presented in Section 8.2, including in particular an IQA pipeline implementation and a prototypical user interface adopted in the user evaluation. Note that our approach is independent of any specific implementation of the SQA pipeline formalized in Chapter 4.

8.3.1 IQA Pipeline

The Semantic Question Answering pipeline of IQA instantiated in this work is illustrated in Figure 8.2. This pipeline consists of four components, namely a Shallow Parser, an Entity Linker, a Relation Linker, and a Query Builder.

With the IQA pipeline, we aim to generate several relevant candidate question interpretations to build the interpretation space QIS, to facilitate the user interaction scheme. This method is different

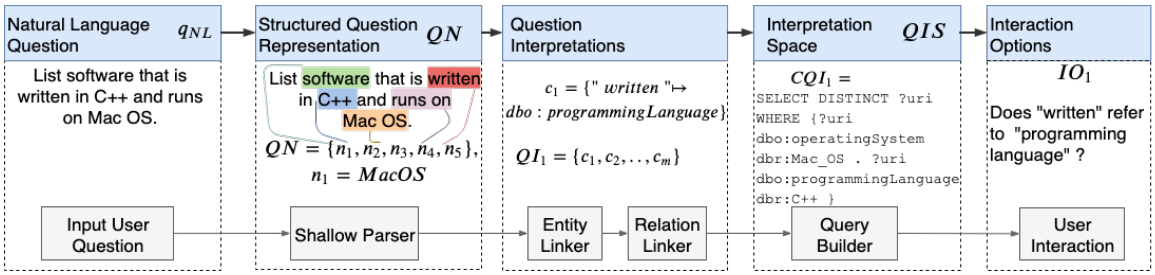


Figure 8.2: An Interactive Question Answering (IQA) pipeline.

from the state-of-the-art SQA approaches such as "WDAqua" [139] aimed to generate only one, the most likely question interpretation.

To increase the recall of relevant question interpretations generated by the IQA pipeline, we leverage multiple independent tools in each pipeline step to obtain complementary candidates. The output of each pipeline component is the union of all candidates produced by the individual tools. This approach increases the recall of the candidates generated in each pipeline component. Furthermore, it increases the overall recall of the relevant question interpretations resulting from the IQA pipeline. To facilitate efficient processing, we run the tools within each pipeline component in parallel.

To select the tools for each pipeline component in the current realization of IQA, we conducted preliminary experiments.

Shallow Parser

We analyzed three independent shallow parsing tools, namely MDP-Parser [140] developed in our previous work, SENNA [71] and a NLTK-based [141] chunker implemented using a classification-based sequential tagger. MDP-Parser is a reinforcement learning-based approach to identify named entity and relation mentions in a distantly supervised setting. In our preliminary experiments, we observed that MDP-Parser shows superior performance for shallow parsing compared to the other approaches [140]. Furthermore, we did not observe any significant performance increase by adopting multiple tools at this pipeline step on the results of the entity and relation linking. Hence, we adopt MDP-Parser as the only tool in the Shallow Parser pipeline component.

Entity Linker

At this stage, we considered two state-of-the-art entity linking tools: TagMe [142] and EARL [70]. To further increase recall, we implemented an additional linking tool, which utilizes a character level n-gram representation of the information nuggets and performs linking between the information nuggets and the labels of entities in the knowledge graph using 3-gram similarity. We implemented this tool using an Apache Lucene index[6]. In our preliminary experiments, we observed that entity linking results obtained by a combination of the 3-gram similarity and EARL subsume the results of TagMe. Thus we adopt the 3-gram similarity and EARL as two independent entity linking tools in the current realization of the IQA pipeline.

[6] https://lucene.apache.org

Relation Linker

Relation linking is conducted analogously to the entity linking, using EARL and a word-matching similarity between the information nuggets and the knowledge graph properties.

Query Builder

We adopt the SQG [110] tool developed in our previous work as the Query Builder component.

8.3.2 Probability Estimation

An estimation of the probability of a complete question interpretation $CQI = (QI, AT, QG)$ presented in Section 4.4, requires estimation of the probabilities of the nugget interpretations $QI = \{ni_1, \ldots, ni_r\}$ and the query graph QG of CQI.

To estimate the probability of a nugget interpretation $P(ni_i|Q, \mathcal{KG})$, we adopt the confidence score of the pipeline component that generates this interpretation. We normalize the confidence scores using min-max scaling.

The probability of the query graph $P(QG|q_{NL}, \mathcal{KG})$ is estimated using structural similarity of the query graph structure QG and the user question q_{NL}. To this extent, we use the Tree-LSTM based model of the SQG tool adopted as the query building component. SQG estimates the syntactical similarity of a candidate query that it generates to the parse tree structure of the input question q_{NL}. To estimate the probability of the query graph, we normalize the similarity score provided by SQG using the softmax function.

8.3.3 IQA User Interface

We implemented IQA prototype as a web application. The user interface of IQA is exemplified in Figure 8.3. This interface is adopted in the user study described later in Section 8.4.4. In general, IQA accepts any user-defined questions in a natural language. To enable a comparison of different approaches in the evaluation, during the user study, we adopted a controlled set of questions selected from the LC-QuAD dataset (we elaborate on the dataset generation later in Section 8.4.1). In this section, we describe the user interface of IQA as it was presented to the users during the user study.

First, the user signs up in the system. Then, the user logs in and starts the user study, where a page similar to Figure 8.3 is presented. At the top of the interface, IQA displays the current question *(#1)*. On the right hand side, the top-ranked query is provided in its natural language representation *(#2)* (using SPARQL2NL [143]) and a SPARQL representation *(#3)*. Using this part of the interface, the user can accept the top-ranked query *(#4)*. Furthermore, if the user finds the presented question or the interaction options incomprehensible, the user can skip the question by choosing the corresponding reason and clicking on the skip button *(#5)*.

On the left-hand side, IQA provides the user with the current interaction option *(#6)*. The interaction option is expressed as an inquiry *(#6.1)* along with a candidate answer *(#6.2)*. The inquiry is in the form of "Does '...' refers to ...?", where '...' is a part of the original question. If applicable, a description and/or example of usages of the interaction option (in case the interaction option represents a relation) are displayed *(#6.3)*. The user can select from "yes", "no", and "I don't know" answers to accept or reject the interaction option displayed *(#6.4)*. The previously selected interaction options are listed below for user reference *(#7)*.

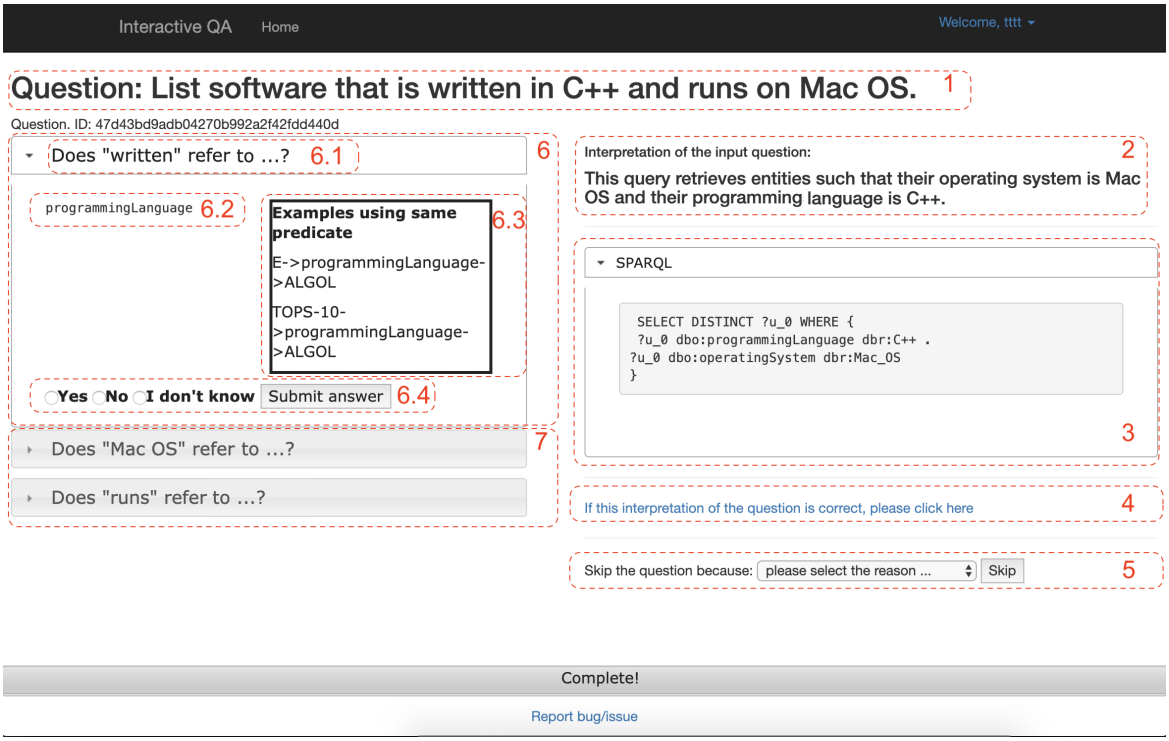


Figure 8.3: User interface of IQA adopted in the user study.

According to the user feedback, the interaction option and the top-ranked query are updated. The interaction continues until the user confirms the final semantic query or another termination criterion discussed in Section 8.2.4 is reached.

To collect the usability feedback, IQA shows a dialog to the user upon completion of each question. In this dialog, IQA asks the user to rate the ease of use of the system. The usability rating is conducted on the scale from one to five, with one being difficult to use and five being easy to use. Finally, IQA presents the user with the next question.

A demo version of the IQA system is publicly available at `http://IQAdemo.sda.tech`.

8.4 Evaluation Setup

The goal of the evaluation is to demonstrate that IQA is competitive compared to both state-of-the-art interactive baselines and non-interactive approaches in terms of the effectiveness, efficiency, and usability for questions of different complexity. In this section, we describe the datasets and methods adopted for the evaluation.

8.4.1 Knowledge Graph and Questions

We adopt LC-QuAD - an established dataset that contains 5,000 complex questions for evaluation of SQA systems [136][7]. Overall, the LC-QuAD dataset contains questions in four complexity categories,

[7] Available at `http://lc-quad.sda.tech/lcquad1.0.html`

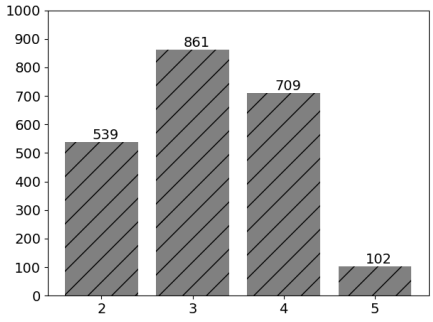


Figure 8.4: Question complexity distribution in the *Oracle Test Questions* dataset. The X-Axis represents the complexity category. The Y-Axis represents the number of questions in the corresponding category.

i.e., questions that include 2-5 named entities and relations in the corresponding semantic queries. Consequently, we use the DBpedia dataset version 2016-10[8] as the underlying knowledge graph to be compatible with the semantic queries in the LC-QuAD dataset.

To the best of our knowledge, Diefenbach et al. [139] provided the state-of-the-art results on the LC-QuAD dataset. Diefenbach et al. use a handcrafted vocabulary expansion for improving relation linking. This vocabulary is based on small parts of training data obtained from various Question Answering datasets, including SimpleQuestions and QALD-7. However, the authors did not clarify whether they use a portion of LC-QuAD to expand the vocabulary, as they do not provide any information regarding the train/test split for LC-QuAD. As the source code of [139] is not available, we used the online API provided by the authors to reproduce their results within each complexity category. We noticed that 2,789 out 5,000 questions in LC-QuAD were not answerable due to the incompatibility of the DBpedia version used for the creation of LC-QuAD and the one used by the API. It was not possible to change the DBpedia version of the API; hence, to provide a fair comparison, we excluded the non-answerable questions and focused on the remaining 2,211 questions. On those questions, our computed F_1 score for WDAqua is 0.438, and their reported score is 0.46, which is similar.

For the oracle-based evaluation, we use the same subset of 2,211 LC-QuAD questions that we used for the evaluation of WDAqua.

We refer to this LC-QuAD subset as *Oracle Test Questions*. Figure 8.4 illustrates the distribution of the questions across the different complexity categories in the *Oracle Test Questions* dataset. As we can observe, the majority of the questions are in the complexity categories from two to four.

For the user evaluation, we select questions for which the IQA pipeline realized in this chapter can generate the semantic query specified in the LC-QuAD dataset (i.e., this query is generated by the IQA pipeline, but is not necessarily top-ranked). From this set, we randomly sample a set of questions, such that the number of questions in each complexity category is balanced. We refer to the set of 90 questions adopted in the user evaluation as *User Test Questions*.

[8] Available at https://wiki.dbpedia.org/downloads-2016-04

8.4.2 Evaluation Metrics

To assess the effectiveness, efficiency, and usability of the considered approaches, we adopt the metrics described in the following.

Effectiveness

To measure effectiveness, we choose *Success Rate* and F_1 *score*.

The *Success Rate* is the percentage of the questions in a dataset for which the SQA approach can generate the intended semantic query. Note, that in case an approach generates several candidates, the intended semantic query does not have to be top-ranked.

The F_1 *score* is the harmonic average of the precision and recall. Here, F_1 score corresponds to the *Success Rate* at the top-1.

Efficiency

To measure efficiency, we adopt *Interaction Cost*. We define the *Interaction Cost* as the number of interaction options that the users need to consider before they can identify the semantic query that correctly interprets the question. In the user evaluation, "identify" means that the user explicitly confirms the semantic query as correct. In the oracle-based evaluation of interaction, "identify" means that the semantic query ranked at top-1 at the specific interaction round corresponds to the query given in the LC-QuAD dataset.

In ranking-based approaches (e.g., in non-interactive baselines), the Interaction Cost is measured as the rank of the correct question interpretation, assuming that the user considers the semantic queries in their rank order.

The lower values of the Interaction Cost correspond to the higher efficiency of an SQA system. The Interaction Cost of '1' corresponds to the case, where the intended semantic query is immediately shown (ranked at top-1) and confirmed by the user.

Usability

To assess usability, we design a rating scheme in which users can provide their feedback on the ease of use on the scale from one to five, with one being difficult to use and five being easy to use.

8.4.3 Evaluated Approaches

In this work, we compare the performance of the SQA approaches and their configurations described in the following.

IQA Configurations

To assess the impact of the Option Gain proposed in this work as opposed to Information Gain, we compare two configurations of the proposed IQA approach: *IQA-OG* and *IQA-IG*.

In *IQA-OG*, the interaction options are selected based on their Option Gain. We set the $\omega = 1$ (see Equation 8.1), such that both, Information Gain and usability of the options are taken into account equally.

IQA-IG is the interactive SQA method, where we take into account the Information Gain of the interaction options only. In this case, we set the parameter $\omega = 0$ (see Equation 8.1).

Baselines

To compare IQA to a state-of-the-art non-interactive SQA approach, we adopt NIB-WDAqua.

NIB-WDAqua: a Non-Interactive SQA Baseline using a state-of-the-art SQA approach. In this case we take the state-of-the-art semantic SQA approach "WDAqua-core1" [139] as a baseline. According to the recent evaluation on the Gerbil platform [144][9], an SQA benchmarking system, "WDAqua-core1" indicates the best performance concerning the LC-QuAD dataset adopted for the evaluation in this chapter. This baseline generates only one semantic query interpreting the user question. This query is provided by the authors of [139] through their API[10].

To demonstrate the performance of the proposed IQA pipeline in the non-interactive settings, we use NIB-IQA.

NIB-IQA: a Non-Interactive SQA Baseline using the IQA pipeline. This baseline represents the IQA pipeline running without interaction. We assume that the IQA pipeline runs entirely automatically and outputs a ranked list of semantic queries at the end, where each semantic query interprets the user question in a specific way. To compute the Interaction Cost for the NIB-IQA baseline, we assume that the user considers the semantic queries generated by the pipeline in their rank order. In this case, the Interaction Cost corresponds to the rank of the semantic query in the resulting list.

To demonstrate the performance of the proposed interaction scheme compared to an interactive baseline, we consider SIB.

SIB: a Simple Interactive Baseline. This baseline involves user interaction after the execution of each SQA pipeline component. We assume that each pipeline component outputs a ranked list of interaction options (e.g., nugget interpretations). Furthermore, the Interaction Cost of each pipeline component is the rank of the first IO generated by this component that leads to the intended semantic query. This option is passed as an input to the next pipeline component. The overall Interaction Cost of the pipeline is the sum of the Interaction Cost over all the pipeline components.

8.4.4 Evaluation Settings

To assess the performance of IQA with respect to the evaluation metrics, facilitate comparison to the baselines and evaluate performance in the interaction involving human users, we performed an oracle-based evaluation and conducted a user study.

Oracle-Based Evaluation

To facilitate evaluation on an established large-scale dataset for Question Answering such as LC-QuAD, we adopt an oracle-based approach.

In particular, in the interaction process, we consider an interaction option to be *correct* if the selection of this option can lead to the construction of the semantic query specified in the LC-QuAD dataset. In the automatic evaluation, we simulate the user interaction process by letting the system automatically accept the first correct option suggested by the adopted SQA method. This corresponds

[9] http://gerbil-qa.aksw.org/gerbil/experiment?id=201805230002

[10] http://wdaqua-core1.univ-st-etienne.fr/gerbil

to the assumption that the user would always select the correct option if this option is suggested by the system.

User Study

To better understand the impact of the proposed Option Gain metric on the effectiveness, efficiency, and usability of the IQA scheme (IQA-OG) in comparison to the interaction based on Information Gain (IQA-IG) when involving human users, we conducted a user study.

To enable evaluation of the proposed approach in the controlled settings, we adopted a homogeneous user group with 15 post-graduate computer science students. We envision evaluation with other user groups to be an important part of the future research.

At the beginning of the study, the users were briefly introduced to the IQA system by the authors. During the study, each user evaluated 12 questions on average (3 questions in 4 complexity categories). On average, users spent 30 minutes to conduct the study. For the configuration of the user study, the following rules were applied:

- To facilitate a comparison of the methods, each question is evaluated using two IQA configurations: IQA-OG and IQA-IG.
- During the study, each user interacts with the system using one fixed interaction configuration, either IQA-OG or IQA-IG.
- The user does not receive the same question twice.
- The user can mark a question as incomprehensible. The question marked by any user is removed from the *User Test Questions* set.

  The remaining set of *User Test Questions* contains 80 questions.

Figure 8.3 illustrates the user interface of IQA adopted in the user study with an example question from the *User Test Questions* set.

User study results are discussed in Section 8.5.2.

8.4.5 Reproducibility

To support the reproducibility of results and facilitate further research, we make the software and the data adopted in the evaluation available as follows. The source code of the interactive query construction is available on our GitHub repository[11]. Similarly, the source code of the MDP-Parser[12], SQG[13] as well EARL[14] are available on GitHub. Furthermore, the experimental results for the oracle evaluation are provided at our GitHub repository [11].

8.5 Evaluation Results

In this section, we present the results of the oracle-based evaluation and the user study.

[11] https://github.com/AskNowQA/InteractiveQA
[12] https://github.com/AskNowQA/DeepShallowParsingQA
[13] https://github.com/AskNowQA/SQG
[14] https://github.com/AskNowQA/EARL

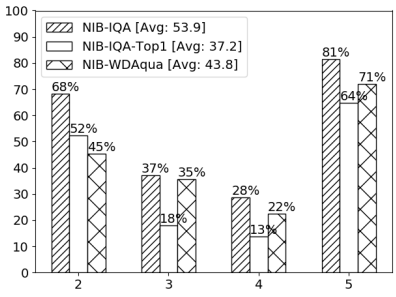


Figure 8.5: Success Rate of the non-interactive baselines NIB-IQA, NIB-IQA-Top-1 and NIB-WDAqua for the questions in the *Oracle Test Questions* dataset. The X-Axis represents the complexity category. The Y-Axis represent the Success Rate.

8.5.1 Oracle-based Evaluation Results

We assess the effectiveness and efficiency of the proposed IQA approach on an established large-scale LC-QuAD dataset using the oracle-based evaluation.

Effectiveness Results in the Oracle-based Evaluation

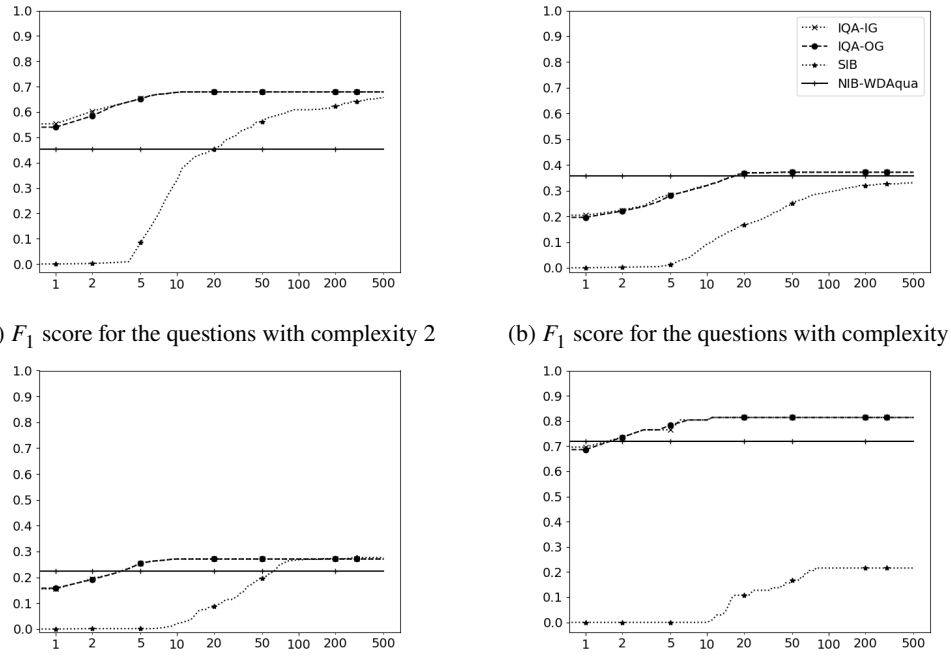
Figure 8.5 presents the Success Rate of the non-interactive baselines. The NIB-WDAqua baseline that represents a state-of-the-art Semantic Question Answering approach [139] generates only the top-1 semantic query. The NIB-IQA baseline, i.e., a non-interactive version of the proposed IQA approach, generates multiple candidate semantic queries. With NIB-IQA-Top-1, we consider only the top-1 query generated by the NIB-IQA baseline.

As we can observe in Figure 8.5, NIB-IQA outperforms the NIB-WDAqua baseline in terms of Success Rate in all complexity categories. Whereas the NIB-WDAqua outperforms the NIB-IQA with respect to the top-1 query (i.e., the NIB-IQA-Top1 baseline), the overall Success Rate of NIB-IQA is higher than the Success Rate of the NIB-WDAqua. This is because NIB-IQA generates multiple relevant question interpretations, whereas NIB-WDAqua does not provide such functionality and returns only one top-ranked query.

As expected, we can observe that the overall performance of all non-interactive question answering pipelines degrades with the increasing complexity of the questions in the categories 2-4. A special case is the Success Rate in the complexity category 5, where the questions follow a similar pattern, which makes it relatively easy for all considered SQA systems to construct the corresponding semantic query.

As we can observe in Figure 8.5, in the complexity category 2, 68% of the queries are answerable by NIB-IQA (i.e., the intended query is constructed by the IQA pipeline), whereas this query is ranked as top-1 (NIB-IQA-Top1) only in 52% of the cases. Overall, the difference between the NIB-IQA and NIB-IQA-Top1 is 16.7 percentage points on average across the complexity categories.

The approach proposed in this chapter fills this gap, such that the difference between NIB-IQA and NIB-IQA-Top1 is reduced through interaction (as will be demonstrated later in the results of the oracle-based evaluation in Section 8.5.1 and the discussion of the user study presented in Section 8.5.2). I.e., with interaction, the Success Rate of the NIB-IQA-Top1 will increase and can reach the Success Rate of NIB-IQA, outperforming the NIB-WDAqua baseline.



(a) F_1 score for the questions with complexity 2

(b) F_1 score for the questions with complexity 3

(c) F_1 score for the questions with complexity 4

(d) F_1 score for the questions with complexity 5

Figure 8.6: Increase in F_1 score during the interaction process in the oracle-based evaluation. The X-Axis represents the number of interactions on a log scale. The Y-Axis represents the F_1 score.

Figure 8.6 shows the F_1 score obtained using different methods and the evolution of the F_1 score during the interaction process achieved due to the reduction of the question interpretation space. The X-Axis represents the number of interactions on a log scale. The Y-Axis represents the F_1 score. We show the results for the questions of different complexity in separate sub-figures of Figure 8.6.

The baseline method NIB-WDAqua conducts only one interaction with the user, i.e., it generates the top-1 semantic query that interprets the question [139]. This semantic query remains unchanged in the interaction process (the API of the [139] does not provide any other interpretations); therefore, the result of the NIB-WDAqua baseline is represented as a straight line in Figure 8.6.

As expected, given the results presented above, the NIB-WDAqua baseline shows the best results at the very beginning of the interaction process in categories 3-5. However, after a few interactions, the NIB-WDAqua baseline is outperformed by other approaches in all complexity categories.

The interactive configurations IQA-OG and IQA-IG of the proposed approach, demonstrate similar performance.

The SIB interactive baseline shows the worst performance across the approaches presented in Figure 8.6 in all complexity categories. SIB implements an extensive interaction strategy and requests user feedback at every pipeline step. This result confirms our intuition that interaction alone is not sufficient to construct the intended question interpretation efficiently. The significant differences between SIB and the informed interaction strategy of IQA (reflected by IQA-OG and IQA-IG) highlight the clear advantage of our proposed approach in comparison to this baseline.

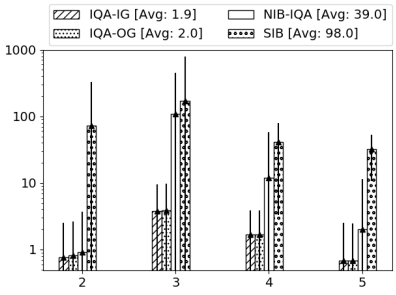


Figure 8.7: Interaction Cost and std. deviation of different approaches in the oracle-based evaluation. The X-Axis represents the complexity category of the question. The Y-Axis represents the Interaction Cost. The Y-Axis is logarithmic. The bars represent the results of the proposed interactive approaches IQA-IG and IQA-OG as well as of the baselines NIB-IQA and SIB.

Efficiency Results in the Oracle-based Evaluation

Figure 8.7 presents the Interaction Cost and the standard deviation of the considered approaches achieved in the different complexity categories in the oracle-based evaluation over the *Oracle Test Questions* dataset.

As we can observe in Figure 8.7, IQA-IG, and IQA-OG have significantly lower Interaction Cost compared to the NIB-IQA and SIB baselines. The Interaction Cost of IQA-OG and IQA-IG in the oracle-based settings are equivalent. This result demonstrates that an interactive approach based on Option Gain or Information Gain can significantly reduce the Interaction Cost compared to the baselines. This result also illustrates that although multiple outputs as produced by the NIB-IQA baseline can facilitate interaction, if taken without further optimization, such multiple outputs are not sufficient to effectively reduce the Interaction Cost.

8.5.2 User Study Results

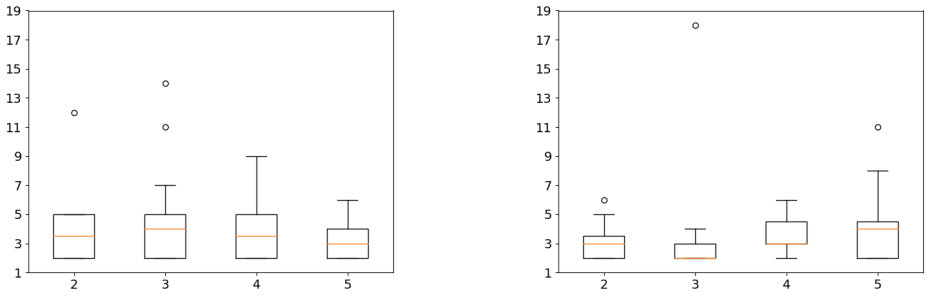
The goal of the user study is to assess the performance of IQA-OG and IQA-IG approaches in terms of their efficiency, usability, and effectiveness in the interaction involving human users. In this section, we present the results of the user study.

Efficiency

We measure the efficiency of interaction using Interaction Cost. Figure 8.8 presents the Interaction Cost observed in the user evaluation for the questions of different complexity while using IQA-OG and IQA-IG configurations of the proposed approach.

Overall, the Interaction Cost of both IQA-OG and IQA-IG is relatively low, with 3.8 interactions on average for IQA-IG and 3.6 for IQA-OG. As we can observe in Figure 8.8, both approaches indicate slight variations. However, the results of the paired t-test show that these differences are not statistically significant. We conclude that both methods, IQA-OG and IQA-IG, are equivalent in terms of efficiency.

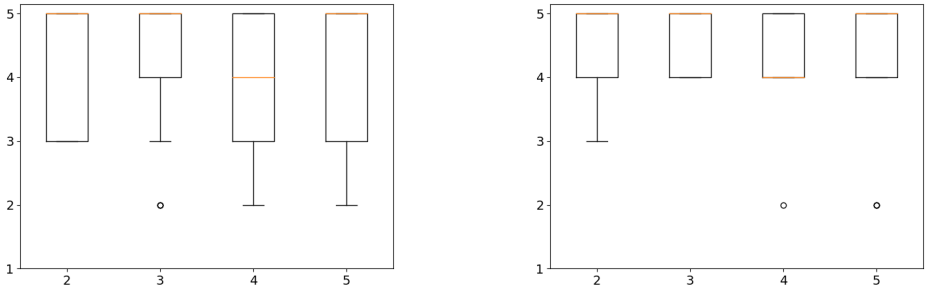
Compared to the results of the oracle-based evaluation, the Interaction Cost observed in the user study is slightly higher. The average Interaction Cost in the oracle-based evaluation presented in



(a) IQA-IG (b) IQA-OG

Figure 8.8: Interaction Cost of IQA-IG and IQA-OG in the user study in a boxplot representation. The X-Axis represents the complexity category. The Y-Axis represents the Interaction Cost.

Figure 8.7 is 1.9-2.0, whereas, in the user study, we observed 3.6-3.8 interactions on average. This is because, in comparison to the oracle-based setting, the users do not always immediately confirm the top-ranked query once it is shown, but may continue the interaction process.



(a) IQA-IG (b) IQA-OG

Figure 8.9: User rating on IQA usability in a boxplot representation. Average rating of IQA-IG=4.13; average rating of IQA-OG=4.40.

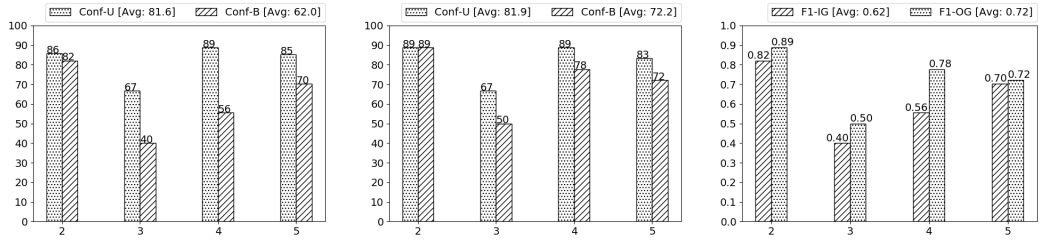
Usability

Figure 8.9 presents the usability results of IQA-IG and IQA-OG computed using user ratings. The average user rating is 4.13 for IQA-IG and 4.40 for IQA-OG. According to the paired t-test, this result is statistically significant ($p < .05$). As we can observe, the scores obtained by IQA-IG are not only lower on average, but also indicate much higher variation. We conclude that IQA-OG outperforms IQA-IG with respect to the ease of use.

Effectiveness

We assess the effectiveness of the interaction scheme in the user evaluation as the accuracy in the construction of the intended semantic queries.

As discussed in Section 8.4.4, to complete the interaction process for each question, the user had to explicitly confirm if the constructed query correctly reflected the intention of the question. The query



(a) IQA-IG (b) IQA-OG (c) F_1 score of IQA-IG and IQA-OG

Figure 8.10: Accuracy of the user judgments vs. the LC-QuAD dataset. The X-Axis represents query complexity. In 8.10(a) and 8.10(b), the Y-Axis represents the ratio of questions for which the semantic query was confirmed by the user (Conf-U) and the ratio of queries, which are equivalent to the LC-QuAD dataset (Conf-B) obtained using IQA-IG and IQA-OG. In 8.10(c), the Y-Axis represents the F_1 score achieved by the users using the IQA-IG and IQA-OG configurations.

confirmed by the user can be different from the semantic query specified in the LC-QuAD dataset. In this section, we discuss the observed deviations between the queries confirmed by the users and the queries specified in the LC-QuAD dataset.

Figures 8.10(a) and 8.10(b) present the ratio of questions in different complexity categories that are: 1) confirmed by the users as correct (Conf-U), and 2) confirmed by the users as correct and also exactly correspond to the semantic query in the LC-QuAD dataset (Conf-B). We present these statistics for the IQA-OG and IQA-IG configurations.

As we can observe in Figures 8.10(a) and 8.10(b), the users have confirmed semantic queries that were not contained in the LC-QuAD dataset in all complexity categories, whereas the differences between Conf-U and Conf-B are much smaller for IQA-OG. Note that Conf-B directly corresponds to the F_1 score presented in Figure 8.10(c).

Figure 8.10(c) indicates that the queries constructed using IQA-OG are more accurate, which is likely due to the interaction options adopted by this approach that can be better understandable by users. The average percentage of queries constructed by the users and confirmed by the LC-QuAD dataset is 62.0% for IQA-IG and 72.2% for IQA-OG. We observe that IQA-OG consistently outperforms IQA-IG in all complexity categories, with an average improvement of 10 percentage points in F_1 score.

This observation again indicates that IQA-OG that takes usability of the options into account can facilitate more effective user interaction than an interaction approach based solely on the Information Gain.

Overall, compared to IQA-IG, IQA-OG leads to more intuitive user interaction that facilitates the user to answer the questions more effectively, within the same number of interactions.

Figure 8.11 depicts the F_1 scores achieved on the *User Test Questions* by different approaches. IQA-IG and IQA-OG scores correspond to the user study results. NIB-WDAqua and NIB-IQA-Top1 are the baseline results achieved on the same dataset. As we can observe, the proposed interactive approach outperforms the best performing non-interactive baseline NIB-WDAqua concerning the F_1 scores in all complexity categories. The average F_1 score of IQA-IG is 0.62, which is an increase of 10 percentage points compared to the NIB-WDAqua baseline that obtains $F_1 = 0.52$ on average on this dataset. With the IQA-OG, we achieve an $F_1 = 0.72$, which is 20 percentage points higher than the F_1 score of the NIB-WDAqua baseline.

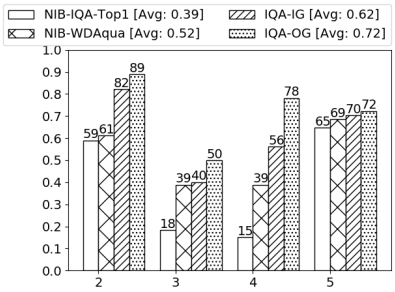


Figure 8.11: The X-Axis represents query complexity. Y-Axis represents the F_1 score achieved by different approaches on the *User Test Questions*. IQA-IG and IQA-OG correspond to the user study results.

Error Analysis

As for the failed questions, on average, 11% were rejected by the users due to incomprehensible questions or interaction options, whereas 15% failed as the users did not confirm the semantic query resulting from the interaction process.

To better understand the differences between the queries constructed and accepted by the users and the semantic queries in the LC-QuAD dataset, we conducted a manual inspection of all results where such deviation occurred. Overall, we observed several reasons for deviations, including:

R1 The LC-QuAD interpretation is too restrictive: There exist several possible semantic interpretations for a question, and LC-QuAD only includes one such interpretation. For example, this can be observed in the case of synonymous relations, or inclusion/omission of the *rdf:type* statements in the semantic query that do not affect the results.

R2 The user makes a mistake or fails to understand the specific differences between the intended interpretation and the interpretation suggested by the system. For example, this can happen in case of similar entities, or a wrong interpretation of the relation direction by the user.

R3 The user selects a different answer type. For example, the user can accept a SELECT query instead of an ASK query specified in LC-QuAD.

We provide an overview of the typical differences, their frequency and the corresponding examples in Table 8.2. As we can observe, the most frequent reasons for the deviations are the synonymous relations (R1, in 43.4%), wrong relations (R2, in 19.5%), and the differences in the answer types (R3, in 19.5%).

User Feedback

After the evaluation session, we requested the users to provide unstructured feedback regarding any issues they observed or comments they had.

Overall, the users reported a positive experience with the IQA system. The typical issues reported by the users included sometimes unclear formulation of the questions in the LC-QuAD dataset, understandability of interaction options in some categories, and of natural language formulation of complex SPARQL queries.

Table 8.2: Differences of the user interpretation and LC-QuAD.

Reason	Differences	%	Example
R1	Synonymous relations	43.4	Q: Name the home stadium of FC Spartak Moscow? dbp:stadium vs. dbo:homeStadium
R1	Completeness of the semantic query	8.6	Q: Miguel de Cervantes wrote the musical extended from which book? SELECT ?u WHERE { ?u dbo:author dbr:Miguel_de_Cervantes } SELECT ?u WHERE { ?u dbo:author dbr:Miguel_de_Cervantes. ?u rdf:type dbo:Book }
R2	Similar entities	4.5	Q: In which state is Red Willow Creek? dbr:Willow_Creek_mine vs. dbr:Red_Willow_Creek
R2	Wrong relation	19.5	Q: List the producer of the TV shows whose company is HBO. dbo:distributor vs. dbo:company
R2	Structural differences in the semantic query	4.5	Q: Who are the predecessors of John Randolph of Roanoke? SELECT ?u WHERE { dbr:John_Randolph_of_Roanoke dbp:predecessor ?u} SELECT ?u WHERE { ?u dbp:predecessor dbr:John_Randolph_of_Roanoke}
R3	Differences in the answer type	19.5	SELECT ?u WHERE ... vs. ASK WHERE ...

As reported by the users, the LC-QuAD dataset contains some questions with linguistic issues. In cases where these issues affected the understandability of questions, the users could skip the question, as mentioned above. We consider such questions as failed in our results.

The users also reported occasional difficulties in understanding the semantics of some of the interaction options, in particular concerning the options representing relations and question types. This observation confirms our assumption used as a basis for the Option Gain computation that the usability of different interaction option types varies.

Finally, the users reported that some of the natural language representations of the SPARQL queries, especially in the context of the more complex questions, were difficult to understand. The generation of the natural language representations for the user interface is not in the scope of this work; in the IQA prototype implementation, we generated such representations using state-of-the-art tools. However, this observation indicates the need for future work in this area.

8.6 Related Work

Interactive methods to obtain user feedback have been adopted in Semantic Question Answering systems as well as in keyword search and natural language interfaces for structured data. In this section, we briefly summarize the differences between IQA and these approaches.

8.6.1 Interactive Keyword Search over Relational Data

In our previous work we proposed FreeQ - an interactive keyword search approach for relational databases [137],[145],[146]. FreeQ generates interaction options using a relational database schema and a mapping between the schema and an external ontology (utilizing, e.g., YAGO+F [147]). User interaction in FreeQ is based on Information Gain. Whereas IQA builds upon our previous work in the area of interactive keyword search, in this chapter, we target a more complex problem of Semantic

Question Answering. The input questions are more complex than keyword queries supported by FreeQ, so are the corresponding SQA pipelines. IQA addresses these challenges through a novel interaction scheme dedicated to Semantic Question Answering. In particular, in IQA, we developed an interaction scheme for generic Semantic Question Answering pipelines. Furthermore, we introduced the notion of Option Gain that takes the usability of interaction options into account. As our evaluation demonstrates, these contributions lead to significant improvements in terms of usability and effectiveness, while maintaining low interaction cost.

8.6.2 Semantic Question Answering

Semantic Question Answering over knowledge graphs is a difficult problem [10, 14]. Although SQA systems over simple questions have improved in recent years [51, 116], solving complex questions [48, 136, 148] remains a difficult task. For example, the "WDAqua-core1" system [139], currently the best performing over the LC-QuAD dataset containing complex queries, only achieves $F_1 = 0.46$. SQA systems usually suffer a performance loss due to the wrong interpretations during the entity linking [70, 142], relation linking, and query building [110] stages. These systems are typically optimized to produce one intended interpretation. In contrast to IQA, such systems do not support user feedback to refine their results.

8.6.3 Interactive Question Answering Systems

Existing SQA and search systems over knowledge graphs employ user feedback and additional input to improve disambiguation of the questions directly, or to generate training data. For example, Exemplar Queries [149] employs a user query as an example to search for similar structures. Su et al. [150] exploit relevance feedback to tune ranking functions in knowledge graph search. GQBE [13] takes a question and an example relation as input and searches for similar graph patterns. Zheng et al. [151] conduct interactive graph search and let users verify the ambiguities in entity linking, relation linking, and query building. IMPROVE-QA [12] asks the users to correct the output of the training question to improve relation linking and query building process by learning from user interaction. In contrast to existing SQA systems that adopt interaction, IQA explicitly addresses the usability aspects of user interaction through Option Gain and utilizes a broader range of interaction options.

8.6.4 Other Interactive Approaches using Knowledge Graphs

Sparklis [152] is an exploration-based approach that allows users to build SPARQL queries interactively. In contrast, IQA is a Semantic Question Answering approach that adopts interaction for the disambiguation of user questions. EventKG+TL facilitates interactive generation of multilingual event timelines from a knowledge graph [153]. Conversational approaches such as CuriousCat [154] provide another type of interaction. These approaches address other objectives, including, for example, knowledge acquisition in a dialog.

8.6.5 Interactive Semantic Parsing

Several works on interactive semantic parsing adopt user feedback as a training signal to resolve utterance ambiguity and enhance parsing accuracy. These approaches translate the natural language to formal domain-specific representations, including database queries [155], API calls [156], and If-Then

programs [157]. Semantic parsing approaches that target translation of natural language into SQL queries for relational databases, such as, for example, [155], [158], [159] are the most related to our work. Approaches in this area are typically limited to rather small database schemas or simple query patterns. For example, [155] performs evaluation on the Microsoft Academic Search (MAS) dataset that includes only eight relations. DialSQL [158] and MISP [159] adopt the WikiSQL dataset that contains rather simple queries. In contrast, interactive SQA systems such as IQA aim to generate semantic queries for knowledge graphs that are much larger in scale, including thousands of concepts and relations, while enabling complex queries. This large scale poses additional challenges concerning the scalability and the interaction cost. Furthermore, approaches to interactive semantic parsing in databases invoke interaction based on ambiguity [155] or error detection [159], and do not address usability aspects.

8.7 Conclusion

In this chapter, we presented our contributions toward **RQ-3**: Exert user interaction to improve QA performance. We introduced IQA - a novel interactive approach to Semantic Question Answering. We proposed a novel probabilistic user interaction scheme. This scheme aims to facilitate the user to effectively identify the intended semantic query while increasing the usability of interaction and minimizing the interaction cost.

Interaction options utilized by the IQA belong to several categories, including interpretations of entities and relations, superclasses and types of entities, answer types, and semantic queries. In the interaction process, these options are determined based on their Option Gain, which takes into account the usability and efficiency of the options.

To evaluate the effectiveness, efficiency, and usability of the proposed user interaction scheme, we conducted an extensive oracle-based experimental evaluation and a user study. Our experimental results over LC-QUAD, an established dataset in the assessment of SQA systems, demonstrate that IQA can significantly increase the effectiveness of SQA for complex questions while maintaining high usability of interaction and incurring only a small interaction cost.

We observed that an interaction strategy IQA-OG based on the Option Gain leads to higher user satisfaction compared to IQA-IG optimized for efficiency only. Furthermore, IQA-OG leads to the higher effectiveness of the user interaction, as reflected by the higher ratio of successfully constructed semantic queries. This improvement is reflected in the F_1 score that outperforms the interaction strategy based on the Information Gain by ten percentage points. Compared to the non-interactive baselines, IQA-OG achieves up to 20 percentage points improvement on the subset of LC-QUAD utilized in the user evaluation. We believe that this improvement is due to the less complex and thus better understandable interaction options adopted by the IQA-OG, which help to reduce potential errors.

In principle, the IQA interaction scheme is applicable on top of any Semantic Question Answering pipeline that realizes the generic architecture formalized in Chapter 4. In particular, we support variations of SQA pipelines in the linking step, such that there can be a single joint linking step for entities and relations or multiple individual linking steps. This way, the IQA interaction approach can be applied to a broader range of existing SQA frameworks.

As noted in the evaluation section, users reportedly found the natural language representation of the formal query hard to comprehend, particularly in the case of complex queries. That, in part, is

the motivation for the next chapter, in which we focus on the last research question **RQ-4**: answer verbalization in semantic question answering.

CHAPTER 9

Answer Verbalization

In the previous chapter, we addressed **RQ-3** by introducing a generic framework to integrate user interaction within the semantic parsing question answering in order to assist the system to capture the correct interpretation of the input question. During the evaluation of the user study, we discover that the natural language representation of the formal query was not always comprehensible, in particular for complex queries.

In addition, given that current question answering systems are not able to be totally certain about their answer, we investigate, through a user study, the effectiveness and usability of the existing representations as well as a new answer verbalization (**RQ-3**). Based on the promising results of the answer verbalization representation from the user study, we introduce the task of answer verbalization: The goal is to provide a large scale resource in order to train machine learning based models to generate natural language representation of the formal query and the answer. This enables the question answering system to go beyond just providing the answer to the users as an isolated fact, but in form of a full sentence in which, the required building blocks from the underlying knowledge graph as well as other features from the formal query is also presented to the users. This presents the users with the necessary information to validate the answer.

In the following section, we focus on the definition of answer verbalization and its challenges. Section 9.2 presents the impact of our dataset within the QA community and its differences in comparison to the existing QA datasets. We introduce the details of our dataset and the generation workflow in Section 9.3. Section 9.4 discusses availability of the dataset. We present the user study setup and its results in Section 9.5, followed by the reusability study in Section 9.6, and Section 9.7 concludes our contributions.

9.1 Introduction

The early Knowledge Graph based Question Answering (KGQA) systems were mostly template or rule-based systems with limited learnable modules [48, 49], mainly due to the fact that the existing QA datasets were small-scaled [50]. Consequently, researchers in the QA community are working on expanding QA datasets from two perspectives: (i) size: to support machine learning approaches that need more training data [51] and (ii) complexity: to move on from simple factoid questions to complex questions (e.g. multi-hop, ordinal, aggregation, etc) [52]. Note that while there are some QA datasets that are automatically generated [53], most QA datasets are manually created either by

(i) using in-house workers [54] or crowd-sourcing [55] (ii) or extract questions from online question answering platforms such as search engines, online forum, etc [52]. The goal is to create datasets that are representative in terms of the types of questions that users are likely to ask.

These large-scale and complex QA datasets enable researchers to develop end-to-end learning approaches [8] and support questions with various features of varying complexity [20]. As a result, the main focus of many competitive QA methods is to enhance the performance of QA systems in terms of the accuracy of answer(s) retrieval. However, the average accuracy of the current state of the art QA approaches on manually created QA datasets is about 0.49, hence, there is plenty of room for improvement (See Figure 9.1).

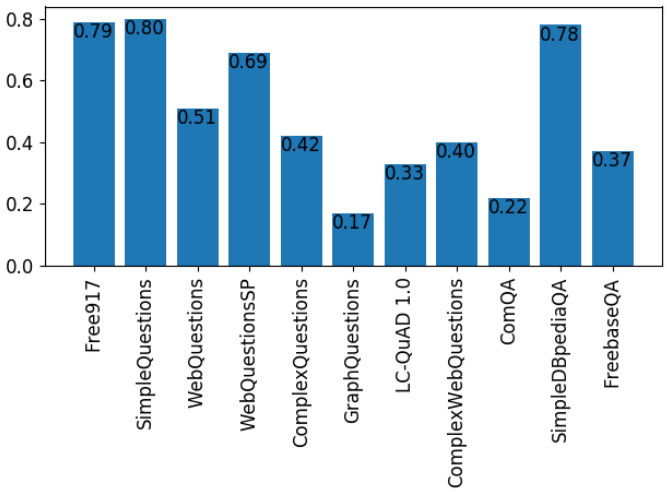


Figure 9.1: The accuracy of the state of the art QA over KGs systems

Consequently, given this accuracy, the answers provided by a QA system need to be validated to assure that the questions are understood correctly and that the right data is retrieved. For instance, assuming that the answer to the exemplary question *"What is the longest river in Africa?"* is not known by the user. If the QA system only provides a name with no further explanation, the user might need to refer to an external data source to verify the answer. In an attempt to enable the users to verify the answer provided by a QA system, researchers employ various techniques such as (i) revealing the generated formal query [160], (ii) graphical visualizations of the formal query [161] and (iii) verbalizing the formal query [143, 162, 163]. We take a different approach to addressing the problem of validating the answers given by the QA system. We aim to verbalize the answer in a way that it conveys not only the information requested by the user but also includes additional characteristics that are indicative of how the answer was determined. For instance, the answer verbalization for the example question should be *"The longest river in Africa is Nile"* and given this verbalization, the user can better verify that the system is retrieving a *river* that is the *longest* river, which is located in *Africa*. Note that though it resembles the idea of verbalization of the formal query, the key difference lies in the fact in answer verbalization, the answer is a full sentence with all the context information that is captured by the formal query; while, the answer plays no role in verbalization of the formal query. Furthermore, answer verbalization can be used to mimic the seamless human conversation.

We design a user study to evaluate the existing approaches to the the problem of verifying the

answers provided by the QA system as well as the verbalized representation of the answer, in terms of accuracy, processing time and ease-of-use. Our findings from the user study suggest that verbalized answer, when correctly and soundly provided, achieves remarkable success in compare to other approaches. (See Section 9.5)

In this context we make the following contributions:

- We provide a framework for automatically generating the verbalization of answer(s), given the input question and the corresponding SPARQL query, which reduces the needed initial manual effort. The questions generated by the framework are subsequently manually verified to guarantee the accuracy of the verbalized answers.
- We present VQuAnDa – Verbalization QUestion ANswering DAtaset – the first QA dataset, which provides the verbalization of the answer in natural language.
- Evaluation baselines, based on a set of standard sequence to sequence models, which can serve to determine the accuracy of machine learning approaches used to verbalize the answers of QA systems.

The further advantages of having a dataset with accurate verbalization of the answer are multi-fold. Users do not need to understand RDF/ the formalization of the results, which decreases the adoption barriers of using KGs and QA systems. In addition, by providing indications of how the answer was derived as part of the verbalization, we enhance the explainability of the system. Furthermore, VQuAnDa serves as the basis for training and developing new ML models, which was up to date difficult due to the lack of data. Finally, our dataset lays the foundation for new lines of work towards extending VQuAnDa by the community.

9.2 Impact

Question Answering (QA) datasets over Knowledge Graphs (KG) commonly contain natural language questions, corresponding formal queries and/or the answer(s) from the underlying KG. Table 9.1 summarizes the features of all existing QA datasets over KGs. All QA datasets (except for [53]) are created using human annotators to ensure the quality of the results. In some datasets (such as FreebaseQA [64] and Free917 [50]), the questions are collected from search engines or other online question answering platforms and were subsequently adapted to an open-domain knowledge graph by human annotators. Others formulate the questions from a list of keywords (for instance LC-QuAD 1.0 [54]), or compose the question given a template-based automatically generated pseudo-question (for instance LC-QuAD 2.0 [55]).

The general trend in QA datasets is to work on the following aspects: (i) to increase the size of the dataset, (ii) to expand the question types to cover various features such as boolean queries, aggregations, ordinals in queries, etc. (iii) to increase the complexity of question by using compound features such as comparison or unions. Most recently, in an attempt to provide human-like conversations on a single topic, researchers expanded QA datasets to cover multiple utterances turns by introducing CSQA [63] – a sequential QA dataset in which instead of isolated questions, the benchmark contains a sequence of related questions along with their answers. However, the dataset contains only plain answers with no further verbalization to mimic human conversation.

Table 9.1: Summary of QA datasets over knowledge graphs

Dataset	KG	Size	Year	Formal Rep.	Creation
Free917 [50]	Freebase	917	2013	SPARQL	Manual
WebQuestions [52]	Freebase	5810	2013	None	Manual
SimpleQuestions [51]	Freebase	100K	2015	SPARQL	Manual
WebQuestionsSP [57]	Freebase	5810	2016	SPARQL	Manual
ComplexQuestions [58]	Freebase	2100	2016	None	Manual
GraphQuestions [59]	Freebase	5166	2016	SPARQL	Manual
30M Factoid Questions [53]	Freebase	30M	2016	SPARQL	Automatic
QALD (1-9) [1]	DBpedia	500	2011-2018	SPARQL	Manual
LC-QuAD 1.0 [54]	DBpedia	5000	2017	SPARQL	Manual
ComplexWebQuestions [60]	Freebase	33K	2018	SPARQL	Manual
ComQA [61]	Wikipedia	11K	2018	None	Manual
SimpleDBpediaQA [62]	DBpedia	43K	2018	Inferential Chain	Manual
CSQA [63]	Wikidata	200K	2018	Entities/Relations	Manual
LC-QuAD 2.0 [55]	Wikidata	30K	2019	SPARQL	Manual
FreebaseQA [64]	Freebase	28K	2019	Inferential Chain	Manual

On the one hand, recent advances in task-oriented dialogue systems resulted in releasing multi-turn dialogue datasets that are grounded through knowledge bases [164]. The intention is to provide training data for models allowing them to have a coherent conversion. However, users cannot validate whether the provided answer at each step is correct. Moreover, the underlying knowledge graphs are significantly smaller than open-domain knowledge graphs such as DBpedia in terms of the number of entities and relations.

Considering the existing QA datasets and task-oriented dialogue datasets, we observe that the verbalization of answers with the intention to enable the users to validate the provided answer is neglected in the existing datasets. Consequently, the existing works cover either (i) the verbalization of the answer as in the dialog dataset, however, without empowering the users to validate the answer, (ii) or they enable the user to validate the answer, however, without a human like conversation [163].

We fill this gap in the question answering community by providing VQuAnDa, thus facilitating the research on semantic-enabled verbalization of answers in order to engage the users in human-like conversations while enabling them to verify the answers as well. We provide the verbalization of the answers by compiling all the necessary elements from the formal query and the answers into a coherent statement. Given this characterization of the answer, the user is enabled to verify whether the system has understood the intention of the question correctly. Furthermore, the dataset can be beneficial in dialog systems to not only hold the conversation but also to augment it with relevant elements that explain how the system comprehends the intention of the question.

We provide details on the dataset and the generation workflow in the next section.

Table 9.2: Examples from VQuAnDa

Question	What is the common school of Chris Marve and Neria Douglass?

Query	SELECT DISTINCT ?uri
	WHERE {
	dbr:Chris_Marve dbo:school ?uri .
	dbr:Neria_Douglass dbo:almaMater ?uri .
	}

Query result	dbr:Vanderbilt_University
Verbalization	[Vanderbilt University] is the alma mater of both Chris Marve and Neria Douglass.
Question	List all the faiths that British Columbian politicians follow?

Query	SELECT DISTINCT ?uri
	WHERE {
	?x dbp:residence dbr:British_Columbia .
	?x dbp:religion ?uri .
	?x a dbo:Politician .
	}

Query result	dbr:{Anglican, Anglicanism, Catholic Church, United Church of Canada, Fellowship of Evangelical Baptist Churches in Canada, Mennonite Brethren Church, story.html, Sikh, Roman Catholic}
Verbalization	The religions of the British Columbia politicians are [Anglican, Anglicanism, Catholic Church, United Church of Canada, Fellowship of Evangelical Baptist Churches in Canada, Mennonite Brethren Church, story.html, Sikh, Roman Catholic].

9.3 VQuAnDa: Verbalization QUestion ANswering DAtaset

We introduce a new dataset with verbalized KBQA results called VQuAnDa. The dataset intends to completely hide any semantic technologies and provide a fluent experience between the users and the Knowledge Base. A key advantage of the verbalization is to support the answers given for a question/query. By receiving a complete natural language sentence as an answer, the user can understand how the QA system interpreted the question and what is the corresponding result. Table 9.2 shows some verbalization examples from our dataset. In the first example, the question *"What is the common school of Chris Marve and Neria Douglass?"* is translated to the corresponding SPARQL query, which retrieves the result `dbr:Vanderbilt_University` from the KB. In this case, the full verbalization of the result is *"[Vanderbilt University] is the alma mater of both Chris Marve and Neria Douglass."*. As it can be seen, this form of answer provides us the query result as well as details about the intention of the query.

Our dataset is based on the Largescale Complex Question Answering Dataset (LC-QuAD), which is a complex question answering dataset over DBpedia containing 5,000 pairs of questions and their SPARQL queries. The dataset was generated using 38 unique templates together with 5,042 entities and 615 predicates. To create our dataset, we extended the LC-QuAD by providing verbalizations for

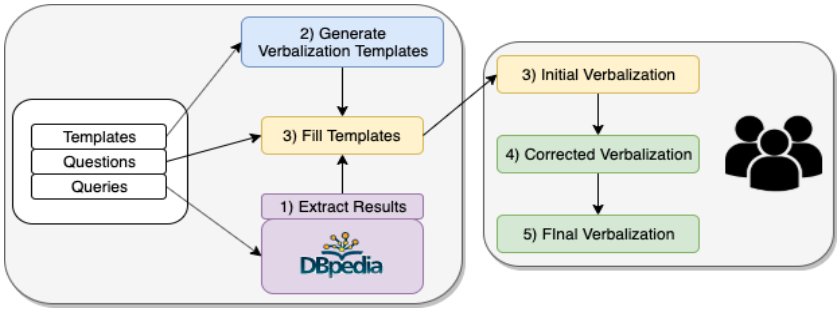


Figure 9.2: Overview of dataset generation

all results. Furthermore, we improved the quality of the dataset by fixing grammar mistakes in the questions and, in some cases where the wording was unclear, completely rewriting them.

Given that Freebase is no longer publicly maintained, we decided to focus on the QA datasets that are based on other KGs such as DBpedia or Wikidata. Therefore, QALD, LC-QuAD 1.0, LC-QuAD 2.0 and CSQA are the only viable options. However, the size of the QALD dataset is significantly smaller in comparison to the other datasets (See Table 9.1). Moreover, in contrast to LC-QuAD 2.0 and CSQA that have not been yet used by any QA system, LC-QuAD 1.0 was the benchmarking dataset in more than 10 recent QA systems. Thus, we choose to built our dataset over LC-QuAD because of the large variety of questions and the manageable size that it has, which allows us to estimate the effectiveness of the produced results.

9.3.1 Generation Workflow

We followed a semi-automated approach to generate the dataset. The overall architecture of the approach is depicted in Figure 9.2.

Extract Results and Set Limit Initially, we retrieved the answers to all questions by using the DBpedia endpoint. Since some questions had multiple results, we had to set a limit on how many answers will be shown as part of the verbalization. In the dataset, there are questions with one result and others with thousands. Creating a verbalization with a long list of all results is not intuitive, often not readable, and it is not contributing to the main focus of our work. Therefore we set a limit of a maximum of 15 results that are shown as part of the verbalization sentence. This limit was chosen by considering the different types of questions within the dataset and their complexity. In *Section 3.2 Statistics* we provide further details about the characteristics of the results. To handle the cases with more than 15 results we decided to replace them with an answer token ($[answer]$). For instance, for the question *"Which comic characters are painted by Bill Finger?"* the corresponding query retrieves 23 comic characters, therefore, the verbalization will include the answer token and it will be *"Bill Finger painted the following comic characters,* $[answer]$*."*. In this way, we can guarantee the sentence fluency for the particular example and we can still consider it for the verbalization task.

Generate Verbalization Templates Next, we generated the templates for the verbalized answers. In this step, we used the question templates from the LC-QuAD dataset. We decided to paraphrase them

using a rule-based approach (see *Section Verbalization Rules*) and generate an initial draft version of the verbalizations.

Create Initial Verbalization In the following, we filled the templates with entities, predicates, and query results. To be able to distinguish the query results from the remaining parts of the verbalization sentence we decided to annotate them using box brackets. This provides us the flexibility whether we want to include, cover or exclude the results while working with the dataset.

Correct and Final Verbalizations While all initial draft versions of the verbalized answers were automatically generated, the last 2 steps had to be done manually in order to ensure the correctness of the verbalizations. First, we corrected and, if necessary, rephrased all answers to sound more natural and fluent. Finally, to ensure the grammatical correctness of the dataset, we peer-reviewed all the generated results.

Verbalization Rules

During the generation workflow, we followed 4 rules on how to produce proper, fluent and correct verbalizations. These rules are:

- Use active voice;
- Use paraphrasing and synonyms;
- Construct the verbalization by using information from both the question and the query;
- Allow for rearranging the triple's order in verbalization.

The first and most important rule is the use of active voice as much as possible. In this way, we produce clean results that are close to human spoken language. The second rule is to paraphrase the sentences and use synonyms for generating different alternatives. The third rule is to base the verbalization on both questions and queries. We have many examples where the question is not directly related to a query from the aspect of structure and words it uses. During the process, we tried to balance out this difference by creating verbalizations that are closer to one or both of them. The last rule enables us to be flexible with the structure of the sentence. We tried not to directly verbalize the triple structure referred to by the query but also to shift the order of the subject and object in order to create more natural sounding sentences. All the rules have been heavily considered during the manual steps. For the automatic template generation, we mostly considered the first and last rule (active voice and sentence structure).

9.3.2 Statistics

In this section we provide more details on the data contained in VQuAnDa, specifically focusing on the distribution of the query results. The dataset consists of 3053 (61.1%) questions that retrieve only one result from the knowledge base. These examples include also boolean and count questions. There are 1503 (30.1%) examples that have more than one answer but less or equal to 15, which is the maximum number that we display as part verbalization. Finally, only 444 (8.9%) examples have more than 15 answers and are replaced with an answer token. Figure 9.3 depicts the result distribution.

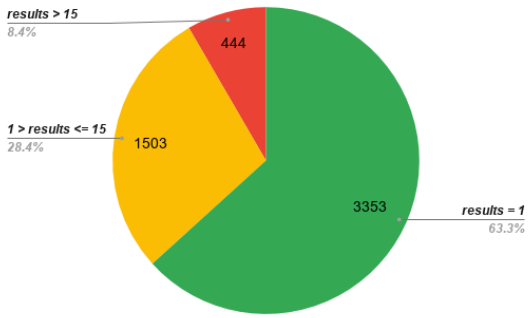


Figure 9.3: Number of Results Returned per Query

Regarding the modified questions in the dataset – 340 (6.8%) examples in the LC-QuAD were revised to better represent the intention of the query. Some of the modifications are grammatical mistakes, while for others we had to completely restructure or even rewrite the questions. Figure 9.4 shows the number of modified questions, per question type and modification type.

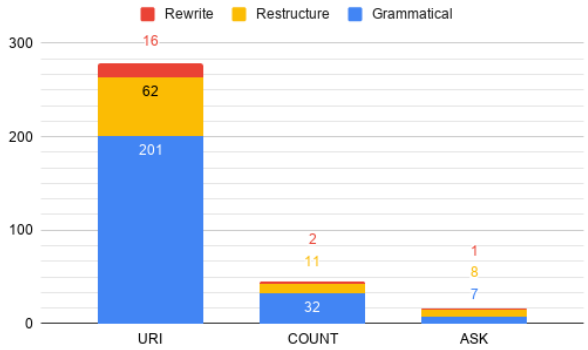


Figure 9.4: Modified Questions in the Dataset

9.4 Availability and Sustainability

The dataset is available at AskNowQA[2] GitHub repository under the Attribution 4.0 International (CC BY 4.0) license. As a permanent URL, we also provide our dataset through figshare at https://figshare.com/projects/VQuAnDa/72488. The repository includes the training and test JSON files, where each of them contains the ids, questions, verbalized answers, and the queries.

The sustainability of the resource is guaranteed by the Question and Answering team of the Smart Data Analytics (SDA) research group at the University of Bonn and at Fraunhofer IAIS. A core team of 3 members is committed to taking care of the dataset, with a time horizon of at least 3 years. The dataset is crucial for currently ongoing PhD research and project work, and will, therefore, be maintained and kept up to date.

[2] https://github.com/AskNowQA/VQUANDA

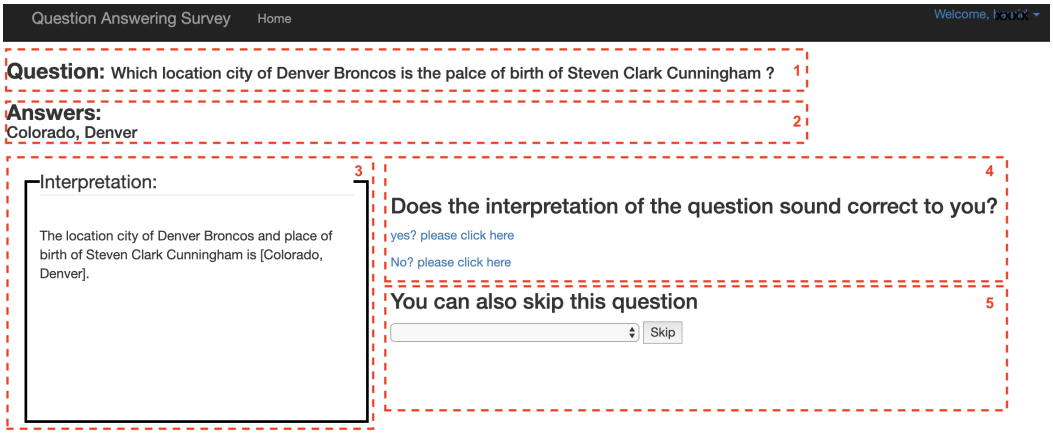


Figure 9.5: The user interface adopted in the user study.

We are planning to have six-months release cycles, regularly updating the dataset based on improvement suggestions and corrections. However, we also plan to further extend VQuAnDa with more verbalization examples. We also aim to make the dataset a community effort, where researchers working in the domain of verbalization can update the data and also include their own evaluation baseline models. VQuAnDa should become an open community effort.

9.5 User study

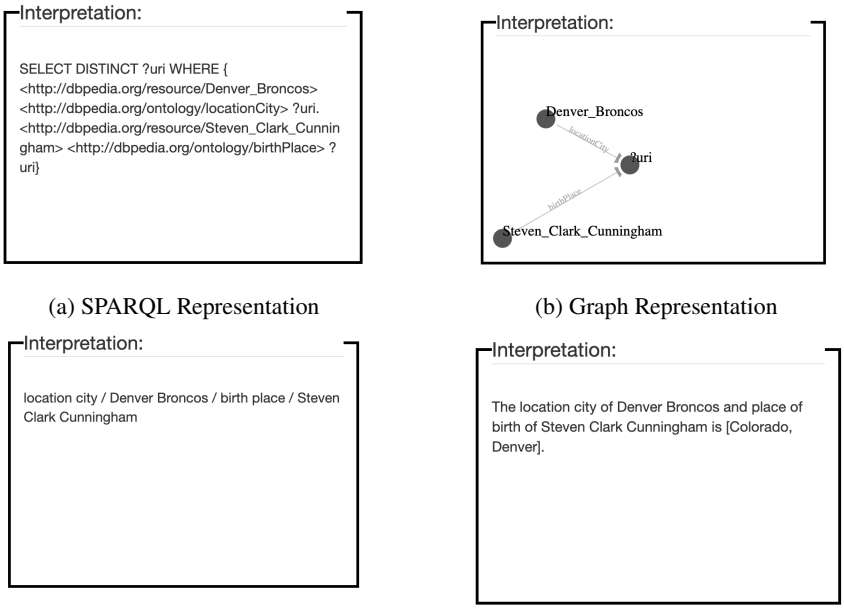
We design a user study to evaluate the performance of various approaches to the problem of validating the answers given by the QA system. This enables us to study users' behavior when they are offered with different representation methods.

In the following, we first describe the user interface which is adopted for the user study. Afterward, we discuss the evaluation setup including evaluation metrics and various approaches that provide a particular representation for the formal query.

9.5.1 User Interface

We develop a web application to carry out the user study. The workflow is as follows: Users need to sign up in the system. Then, users can log in to the system and begins the user study. An explanatory web page is shown to the users to describe the goal of the user study, various steps and the different elements in the user interface. After the brief introduction, users are shown the main page of the user interface (See Figure 9.5). At the top of the user interface *(#1)*, the question is displayed and the answer is presented in the next line *(#2)*. On the left side, the interpretation of the formal query is shown *(#3)*. It is described later in Section 9.5.2. Users can decide whether the interpretation is semantically equivalent to the input question by choosing the corresponding option from the right side of the user interface *(#4)*. Users can also skip the question when the question or its interpretation is not comprehensible by clicking on the skip button on the side right *(#5)*.

Finally, we ask users for their feedback on the usability of the user study on a scale of one to five, with one being hard to use and five being easy to use.



(a) SPARQL Representation

(b) Graph Representation

(c) Controlled Language Representation

(d) Answer Verbalization Representation

Figure 9.6: Various representation approaches of the formal query to the question *"Which location city of Denver Broncos is the place of birth of Steven Clark Cunningham?"*

9.5.2 Evaluation Setup

We adopted a user group of 33 grad and post-grad students (mostly in computer science) and presented them 20 questions that are randomly drawn from the LC-QuAD dataset. The participants were briefly introduced to the systems and they spent 20 minutes on average to conduct the user study. The following rules were observed during the user study:

- Each user is provided consistently with a fixed representation of the formal query.
- No question is provided twice to the same user.
- Users may skip if they find a question or its interpretation as incomprehensible.

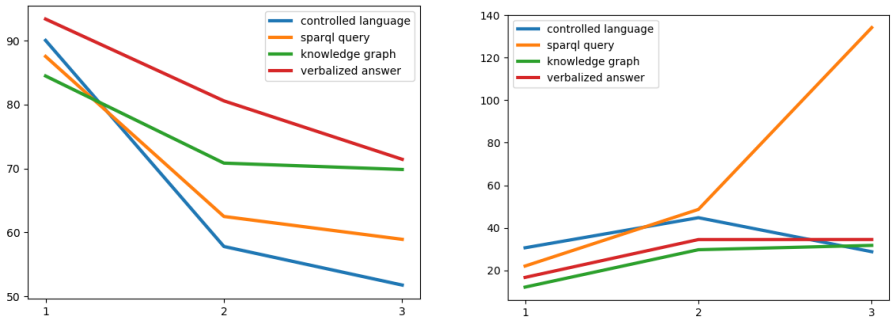
The main page of the user interface is depicted in Figure 9.5.

Metrics

We consider multiple evaluation metrics to capture different aspects: We define **Accuracy** as the percentage of the questions for which users confirm the interpretation of the question and answer. **Duration** specifies the average time that users take to process each question. **Ease of Use** is the explicitly inquired feedback from the users at the end of the user study (on a scale of 1 to 5).

Baselines

We consider the following approaches to provide a different representation of the formal queries to the users:



(a) Accuracy

(b) Average time

- We consider the *Formal representation* as the baseline representation, in which the user is simply provided with a formal query (e.g SPARQL) of the input question, as well as the answer that is retrieved from the knowledge graph. This representation requires certain skills and knowledge about semantic technologies including SPARQL querying language (See Figure 9.6(a)).
- We show a *Graphical interpretation* (similar to [161]) of the formal query that denotes a subgraph from the underlying knowledge graph along with the answer. Figure 9.6(b) depicts an example of this interpretation. While this representation is more intuitive with respect to the previous one, it still requires a limited understanding of knowledge representation with graphs made of nodes and edges. Furthermore, certain formal characteristics such as boolean queries or aggregate functions cannot be integrated into this representation.
- A correct and sound verbalization of the formal query is very challenging. Furthermore, a mistake in the verbalization of the query might confuse the users or change the meaning of it. Inspired by the recent works on verbalizing the formal query [143, 162, 163], we use *controlled language interpretation* as yet another baseline. This interpretation tries to minimize the required skill to understand the interpretation while avoiding the complexity of the complete formal query verbalization. In particular, we benefit from the online API [3] of SPARQLToUser [163] to provide controlled language interpretation. Furthermore, we provide the answer to the user as well (See Figure 9.6(c)).

Moreover, we provide the users with *Answer verbalized representation*, which is based on our hypothesis: to provide a sound and correct verbalization of the answer (see Figure 9.6(d)).

9.5.3 Evaluation Results

We present the result of the user study in this section. The goal of the use study is to analyze the effect of using various interpretation approaches in the performance of the users in terms of accuracy, time and user satisfaction.

Note that we only provide the correct interpretations. While adding incorrect interpretations might be insightful to measure whether users can reject incorrect interpretations, we refrain from doing so to avoid any further complications and focus on measuring the comprehensibility of the interpretations from the user perspective.

[3] https://qanswer-sparqltouser.univ-st-etienne.fr/sparqltouser

While all of the approaches show a declining trend as the complexity increases (See Figure 9.7(b)), users that were shown the *Answer verbalized representation* constantly exhibit better judgments across all the complexities. It was followed by *Graphical interpretation* by a margin of 7%. Figure 9.7(b) unvails that *Formal representation* and *controlled language interpretation* failed to efficiently help users to confirm the interpretation, notably in case more complex questions.

However, Figure 9.7(a) reveals that users with *Answer verbalized representation* spent slightly more (two seconds on average) to provide their feedback, in contrast to the users with *Graphical interpretation*. Our detailed analysis shows the reasons for the low average time in case of the most complex question with *controlled language interpretation* is that users either incorrectly reject the interpretation or choose the interpretation to be incomprehensible. This case is also justified by the very low performance of *controlled language interpretation* in the case of the most complex question in Figure 9.7(a).

Moreover, we computed the average for the ease of use metric: Users with *Answer verbalized representation* assigned the highest value of 4.3 compared to others. It is followed by an average score of 4.0 for *Graphical interpretation*, and 3.8 for *controlled language interpretation* and *Formal representation*.

Our findings from the user study confirm our hypothesis that a correct and comprehensive verbalization of the answer is an effective approach in enabling the users to verify the answer provided for a given question.

9.6 Reusability

The dataset can be used in multiple areas of research. The most suitable one is the knowledge base question answering area, since the initial purpose of the dataset was to support a more reliable QA experience. VQuAnDa allows researchers to train end-to-end models from generating the query, extracting the results and formulating the verbalization answer. Furthermore, the dataset can be used for essential QA subtasks such as entity and predicate recognition/linking, SPARQL query generation and SPARQL to question language generation. These subtasks are already supported by the LC-QuAD dataset. With the verbalizations, researchers can also experiment on tasks such as SPARQL to verbalized answer, question to verbalized answer or even hybrid approaches for generating better results. These possible lines of work indicate that the dataset is also useful for the natural language generation research area.

In summary, the use of the dataset is straightforward and allows researchers to further investigate different fields and discover other possible approaches were KBQA can be done more transparently and efficiently.

9.6.1 Experiments

To ensure the quality of the dataset but also to support its reuse we decided to perform experiments and provide some baseline models. These baseline models can be used as a reference point by anyone working with the dataset.

The experiments are done for the natural language generation task. We would like to test how easy it is for common neural machine translation or sequence to sequence models to generate the verbalized answers using as input only the question or the SPARQL query. To keep the task simple and because

the answers appear only in the output verbalization part, we prefer to hide them with an answer token ($<ans>$). In this way, it will be enough for the model to predict only the position of the answer in the verbalization sentence.

We perform the experiments in two ways – i) the question or SPARQL query will be the input to our models and the expected output will be the correct verbalization; ii) we cover the entities in both input (question or query) and verbalization, so we allow the model to focus on other parts such as the sentence structure and the word relations. For the second experiment approach, we use the EARL framework [70] for recognizing the entities in both the question and answer sentences, and we cover them with an entity token ($<ent>$). For the queries, we can directly cover the entities, since we already know their positions. As we might expect, not all entities will be recognized correctly, but this can happen with any entity recognition framework and especially with datasets with complex sentences that contain one or multiple entities.

In the following subsections, we provide more details about the baseline models, the evaluation metrics, training details, and the results.

Baseline Models

For the baseline models, we decided to employ some standard sequence to sequence models. We first experiment with two RNN models that use different attention mechanisms [165, 166]. For both RNN models we use bidirectional gated recurrent units (Bi-GRU) [167]. Next, we experiment with a convolutional sequence to sequence model, which is based on the original approach [168] where they employ a convolutional neural network (CNN) architecture for machine translation tasks. Finally, we use a transformer neural architecture, which is based on the original paper [169] where they create a simple attention-based sequence to sequence model.

Evaluation Metrics

BLEU: The first evaluation metric we use is the Bilingual Evaluation Understudy (BLEU) score introduced by [170]. The idea of the BLEU score is to count the n-gram overlaps in the reference; it takes the maximum count of each n-gram and it clips the count of the n-grams in the candidate translation to the maximum count in the reference. Essentially, BLEU is a modified version of precision to compare a candidate with a reference. However, candidates with a shorter length than the reference tend to give a higher score, while candidates that are longer are already penalized by the modified n-gram precision. To face this issue a brevity penalty (BP) was introduced, which is 1 if the candidate length c is larger or equal to the reference length r. Otherwise, the brevity penalty is set to $\exp(1 - r/c)$. Finally, a set of positive weights $\{w_1, ..., w_N\}$ is determined to compute the geometric mean of the modified n-gram precisions. The BLEU score is calculated by:

$$BLEU = BP \cdot \exp(\sum_{n=1}^{N} w_n \; log \; p_n), \tag{9.1}$$

where N is the number of different n-grams. In our experiments, we employ $N = 4$ and uniform weights $w_n = 1/N$.

Perplexity: To estimate how well our models predict the verbalization we are using the perplexity metric. For measuring the similarity of a target probability distribution p and an estimated probability distribution q, we are using cross entropy $H(p, q)$ which is defined by

$$H(p,q) = -\sum_{x} p(x)\ log\ q(x), \tag{9.2}$$

where x indicates the possible values in the distribution. The perplexity is defined as the exponentiation of cross entropy:

$$Perplexity(p,q) = 2^{H(p,q)}. \tag{9.3}$$

In our case, the target distribution p is the encoding vector of the verbalization vocabulary and q is the prediction output of the decoder. We calculate perplexity after every epoch using the averaged cross entropy loss of the batches. Researchers have shown [171] that perplexity strongly correlates with the performance of machine translation models.

Training

To keep the comparison fair across the models we employ the same training parameters for all. We split the data into 80-10-10 where 80% is used for training, 10% for validation and the last 10% for testing. The batch size is set to 100 and we train for 50 epochs. During the training, we save the model state with the lowest loss on the validation data.

We tried to keep the models almost of the same size regarding their trainable parameters. More precisely, for the first RNN model we use an embedding dimension of 256, the hidden dimension is 512 and we use 2 layers. We also apply dropout with probability 0.5 on both encoder and decoder. For the second RNN model, we keep everything the same except the embedding dimension where we decided to double it to 512. For the convolutional model, we set the embedding dimension to 512, we keep all the channels in the same dimension of 512 and we use a kernel size of 3. We use 3 layers for the encoder and decoder. Similar to RNNs, the dropout here is set to 0.5. Finally, for the transformer model, the embedding dimension is 512, we use 8 heads and 2 layers. The dropout here is set to 0.1. For the first two RNN models we use a teacher forcing value of 1.0 so we can compare the results with the other models.

We do not use any pretrained embedding model. For building the vocabularies we use a simple one-hot encoding approach. For all the models we use Adam optimizer, and cross entropy as a loss function. All our experiments are publicly available here https://github.com/endrikacupaj/VQUANDA-Baseline-Models.

Results

Beginning with the perplexity results, in Table 9.3 we can see that the convolutional model outperforms all other models and is considered the best. The transformer model comes second and is pretty close to the convolutional. The RNN models perform considerably worse comparing the other two.

Since perplexity is the exponentiation of cross entropy, the lower the value the better the results, which means that the best possible value is 1. The convolutional model using the question as input achieves 4.1 with entities and 3.4 with covered entities on validation and test data. When we use the SPARQL query as input the perplexity gets a lower value, which means the model performs slightly better. In particular, for the convolutional model, we obtain 3.3 with entities and 3.2 with covered entities on test data. The improved performance using the query as input is expected since the model receives the same pattern of queries every time.

Table 9.3: Perplexity experiment results

Models	Input	With Entities		Covered Entities	
		Validation	Test	Validation	Test
RNN-1 [165]	Question	8.257	8.865	5.709	5.809
	Query	6.823	7.029	5.212	5.335
RNN-2 [166]	Question	8.494	8.802	5.799	5.891
	Query	6.727	6.999	5.259	5.394
Convolutional [168]	Question	**4.137**	**4.194**	**3.409**	**3.451**
	Query	**3.175**	**3.311**	**3.158**	**3.201**
Transformer [169]	Question	5.232	5.464	3.716	3.727
	Query	3.978	4.062	3.229	3.292

Table 9.4: BLEU score experiment results

Models	Input	With Entities		Covered Entities	
		Validation	Test	Validation	Test
RNN-1 [165]	Question	14.00	12.86	25.09	24.88
	Query	18.40	17.76	30.74	29.25
RNN-2 [166]	Question	15.53	15.43	27.63	26.95
	Query	22.29	21.33	**34.34**	30.78
Convolutional [168]	Question	**21.49**	**21.30**	**28.21**	**27.73**
	Query	**26.02**	**25.95**	32.61	**32.39**
Transformer [169]	Question	19.00	18.38	25.67	26.58
	Query	24.16	22.98	31.65	29.14

In any case, there is still a lot of space for improvement until we can say that the task is solved. The BLEU score further supports this fact. By looking at Table 9.4 with the BLEU score results we can see that again the convolutional model performs best with a value of up to 32 with covered entities. Without covering entities the best we get on test data is almost 26, which is not really an adequate result. The best possible value for the BLEU metric is 100. A score of more than 60 is considered a perfect translation that often outperforms humans. For our dataset, there is still a lot of research required until we produce models that can reach these numbers.

9.6.2 Use by the Community

Currently, we are actively developing and sharing the dataset within the scope of two projects – SOLIDE and CLEOPATRA. The work is very well received, however, our ultimate goal is to make VQuAnDa an open community effort.

SOLIDE [4] In major disastrous situations such as flooding, emergency services are confronted with a variety of information from different sources. The goal of the SOLIDE project is to analyze and

[4] http://solide-projekt.de

process the information from multiple sources in order to maintain a knowledge graph that captures an overall picture of the situation. Furthermore, using a voice-based interface, users can ask various questions about the ongoing mission, for instance, *"How many units are still available?"*. Given the dangerous circumstances, it is vital to assert the user that the provided answer is complete and sound. Hence, the system needs to verbalize its internal representation of the question (e.g. SPARQL) with the answer as *"There are 2 units with the status available"*. Thus, VQuAnDa is essential for being able to learn a verbalization model as part of the solution framework of the SOLIDE project.

CLEOPATRA ITN [5] As European countries become more and more integrated, an increasing number of events, such as the Paris shootings and Brexit, strongly affect the European community and the European digital economy across language and country borders. As a result, there is a lot of event-centric multilingual information available from different communities in the news, on the Web and in social media. The Cleopatra ITN project aims to enable effective and efficient analytics of event-centric multilingual information spread across heterogeneous sources and deliver results meaningful to the users. In particular in the context of question answering, Cleopatra advances the current state of the art by enabling user interaction with event-centric multilingual information. Considering this challenge, the VQuAnDa dataset serves as a basis for learning a question answering model, while the verbalizations are employed to enhance the interactivity of the system.

In addition to using the dataset in order to conduct research and enable the work within projects, we also use it for teaching purposes. VQuAnDa and pre-trained models are given to the students so that they can try out machine learning approaches by themselves and evaluate the produced results by looking at the quality of the generated verbalizations. While the dataset already has a solid level of reuse, we see great potential for further adoption by the Semantic Web community, especially in the areas of applied and fundamental QA research.

9.7 Conclusion

In this chapter, we introduce VQuAnDa – the first QA dataset including the verbalizations of answers in natural language, as an elementary attempt toward the last research question **RQ4**. We complement the dataset by a framework for automatically generating the verbalizations, given the input question and the corresponding SPARQL query. Finally, we also share a set of evaluation baselines, based on a set of standard sequence-to-sequence models, which can serve to determine the accuracy of machine learning approaches used to verbalize the answers of QA systems. Without a doubt, the dataset presents a very valuable contribution to the community, providing the foundation for multiple lines of research in the QA domain.

In the next chapter, we conclude our findings thorough out the thesis with respect to the research questions and provide future directions.

[5] http://cleopatra-project.eu/

CHAPTER 10

Conclusion and Future Directions

In this chapter, we provide a summary of our research work. We organize the chapter with respect to the research questions presented in Chapter 1 and the findings across the thesis. Furthermore, we explore the outlook on future work based on our contributions.

10.1 Research Questions Review

In this thesis, we focused on knowledge based question answering systems. We zoomed into semantic question answering systems. The vast majority of the existing works on question answering introduced a new system from scratch, though with a similar pipeline architecture. This resulted in low reusability in the question answering community. Hence, we aim to address this issue in the first research question:

RQ-1 *How can semantic technologies assist in formalizing semantic question answering?*

We analyzed existing question answering systems and their built-in components. In Chapter 2, we formalized a generic framework to describe each component, their inputs and outputs, and their dependencies. We adhered to the proposed framework in the rest of the thesis, which greatly helped to design new models for various components within the framework. This formalization enabled us to further integrate a user interaction schema within the framework using the existing components from question answering community.

RQ-2 *What are the low performing components in SQA and to what extent can the improvement of bottleneck components in SQA enhance the overall accuracy of the pipeline?*

Given the formalization of semantic question answering, we moved on to discover the components that have received less attention in the community and went on to introduce tailored models to mitigate their low performance. We focused on two components that had been neglected in the semantic question answering community: Shallow Parsing and Query Builder.

In Chapter 5, we reviewed the existing works on the parsing module in semantic question answering systems. We discovered that most of the existing systems either use hand-crafted templates/patterns or borrow exiting tools from the natural language processing field. The first group commonly fails to generalize well and requires manual engineering to be used in other domains. The methods in the second group are commonly trained on large scale corpora of free text that does not necessarily

represent the features of questions (e.g their syntactical structure). Moreover, due to the fact that most existing question answering datasets provide no target annotation for the shallow parsing task, it is not possible to train the existing models using the natural language process. Consequently, we introduced a reinforcement learning based approach that benefited from the distance labels and provided state of the art results on shallow parsing. Through extensive experiments, we showed that it led to a remarkable increase in performance of the linking task and the overall question answering system.

In Chapter 6, we studied the task of query building based on its definition within the proposed and formalized framework. While this is a crucial task in semantic question answering, less attention has been paid to it. This is rooted in the fact that many question answering datasets consist of formal queries with rather simple or limited structure. However, considering the trend in question answering dataset toward multi-hop queries, we proposed a scalable approach toward query building to support single and multi-hop questions, which can be in the forms of count, boolean or list questions. The underlying intuition is that for every question, there exists a corresponding valid walk in the underlying knowledge graph. However, due to the uncertainty that exists in the previous steps, the query builder takes into account the confidence of its input and generates multiple candidate queries. Finally, it uses a Tree-LSTM model that exploits the structural similarity of the candidate queries and syntactical structure of input questions in order to rank the candidate queries. Our experiments revealed the superiority of our proposed model in comparison to the baseline systems.

We further expanded our query builder in Chapter 7 to support more complex cases where the target query has features such as filtering or sorting, by augmenting the query from our existing query builder. Given that these types of questions are not yet very popular in the question answering datasets, existing question answering systems usually handle them via simple ad-hoc techniques such as pattern matching. However, these ideas require manual inspection and pattern extraction for the dataset at hand, thus it is not trivial to use the same approach for other features such as aggregation. We introduced a systematic way to augment the core query by adding a hierarchical classifier to discover the main type of the query as well as its features (filter, ordinal). We conducted comprehensive experiments on several datasets and provided experimental evidence that reveals excellent results in comparison with the baseline system.

RQ-3 *Can user interaction improve the overall accuracy of QA pipelines while maintaining the user satisfaction?* In contrast to the previous research question in which we focused on the individual components to improve, the goal in this research question is to enhance the question answering system as a whole by means of taking into account user interaction. The main intuition comes from the fact that a wrong decision in any of the components may result in the overall failure of the question answering system to capture the correct intention of the question. Hence, we incorporate user feedback to guide the system toward creating a formal representation such that it would correspond to the input questions.

It is vital to construct the correct formal representation of the input question without requiring the user to have any technical knowledge. We tackled this issue by various means in the user interface, for instance by showing the description for an entity, exemplifying a relation, or verbalizing the formal query. However, the main challenge is to balance the trade-off between user involvement and performance. There are already existing works in which integrate user interactions at each step in the question answering pipeline. However, we took a different strategy by computing the whole interpretation space and then exerted a cost-sensitive decision tree to find the optimal interaction

options based on information gain. We further enhanced it by introducing option gain, in which we augmented information gain with its level of complexity (a.k.a usability) to take into account the user perspective to optimize user interactions. This gave us a remarkable advantage in comparison to baseline systems as we were able to find the global optimal interaction option as opposed to the local optimal interaction option at each step.

We conducted oracle evaluations as well as a user study. In oracle experiments, we witnessed a remarkable increase in the performance of our system against the baseline systems on the LC-QuAD dataset. In the user study, we provided a web-based interface to users to guide the system to capture the correct interpretation of the question. We also solicited users for their feedback on the usability of the user interface and interaction options. We observed that the presented option gain led to higher user satisfaction compared to using only information gain.

RQ-4 *Can answer verbalization enable users to verify the provided answer without using any external source of information?*

A natural premise of asking a question is to find the right answer. Most of the existing question answering systems are content to provide the answer, with no further means to enable the users to validate the answer. Mainstream approaches that deal with this problem provide the answer - as well as a more understandable representation of the formal query such as a graph or a controlled language equivalent. In contrast, we believed that it is in essence not compatible with the flow of conversation in human language. Therefore, in order to enable users to verify the answer, we aimed to verbalize the answer in a way that all required information that allows the user to validate the answer is presented. This provides a single coherent sentence to the user that resembles a human-like conversation.

We evaluated our proposed verbalization along with the existing methods to formal query representation (for instance, graphical visualization and controlled language representation) in a user study. Our findings suggest that users achieved better accuracy in a shorter span of time, when they are provided a sound and coherent answer amongst others.

Consequently, We took the first step by providing the required annotation for a commonly used question answering dataset, as well as the baseline models. This ensures that researchers can exert a variety of natural language generation models from various fields to improve the provided baseline models.

10.2 Future Work

In this thesis, we focused on semantic question answering using knowledge graphs. We proposed a generic formalization that clearly defines various components and their dependencies. We studied the existing systems and discovered several gaps. Furthermore, we introduced various models to fill the gaps and increase the overall performance of the question answering system. In addition, we introduced a generic interactive framework that is able to incorporate user feedback to guide a semantic question answering system to capture the correct intention of questions. Finally, we provided a new resource that helps the community to provide answer verbalization to user questions. This resource enables users to receive the answer as a sentence in which s/he can instantly verify the answer without using any external source of information.

There are still many important questions and gaps in semantic question answering. However we would like to focus on more abstract ideas that are mutual with many others in various fields such as

machine learning: During the last decades, the main objective in many fields in computer science, special machine learning, are defined toward finding an approach to maximize a relevant metric. For instance, the main objective of many tasks in machine learning is to find the model that achieves state of the art performance in terms of accuracy or other quantitative measure. As a result, there have been great advancements in black-box methods such as neural networks to set new benchmarks. During the few last years however, concern is increasing in the scientific community over other important considerations such as fairness, ethics and explainability. For instance, studies have shown that patients usually prefer to talk to a specialized individual rather than engaging with a computer program to get advise on health-related issues even if the computer program is more accurate [172].

Hence, the decision making process is not transparent to the users due to lack of any explanation for the decision taken by the computer program. We can adapt the idea for question answering systems: A question answering system that is explainable could be more valued by users than a black-box question answering system with marginally higher accuracy.

We believe the semantic question answering that benefits from structured data such as knowledge graphs, can be tuned to present explanation for their answer via various means. As an example, we provided a resource that opens the way for an explainable answer via answer verbalization in Chapter 9. While it is a first step, we hope that this work will motivate researchers to develop innovative models based on the presented resource.

Bibliography

[1] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. Mendes, S. Hellmann, M. Morsey, P. van Kleef, S. Auer, and C. Bizer, *DBpedia - A Large-scale, Multilingual Knowledge Base Extracted from Wikipedia*, Semantic Web Journal **6** (2015) 167 (cit. on pp. 1, 12, 17, 42).

[2] D. Vrandecic and M. Krötzsch, *Wikidata: a free collaborative knowledgebase*, Commun. ACM **57** (2014) 78 (cit. on pp. 1, 13, 17).

[3] J. Hoffart, F. M. Suchanek, K. Berberich, and G. Weikum, *YAGO2: A spatially and temporally enhanced knowledge base from Wikipedia*, Artif. Intell. **194** (2013) 28 (cit. on pp. 1, 13).

[4] S. Gottschalk and E. Demidova, "EventKG: A Multilingual Event-Centric Temporal Knowledge Graph," *Proceedings of the 15th International ESWC Conference*, 2018 272 (cit. on pp. 1, 13).

[5] S. Gottschalk and E. Demidova, "EventKG - the Hub of Event Knowledge on the Web - and Biographical Timeline Generation," vol. 10, 6, IOS Press, 2019 (cit. on pp. 1, 13).

[6] H. Paulheim, *Knowledge graph refinement: A survey of approaches and evaluation methods*, Semantic Web **8** (2017) 489 (cit. on pp. 1, 13).

[7] K. Höffner, S. Walter, E. Marx, R. Usbeck, J. Lehmann, and A.-C. N. Ngomo, *Survey on challenges of Question Answering in the Semantic Web*, Semantic Web **8** (2017) 895 (cit. on pp. 1, 13).

[8] N. Chakraborty, D. Lukovnikov, G. Maheshwari, P. Trivedi, J. Lehmann, and A. Fischer, *Introduction to Neural Network based Approaches for Question Answering over Knowledge Graphs*, arXiv preprint arXiv:1907.09361 (2019) (cit. on pp. 2, 13, 26, 90).

[9] K. Singh, A. S. Radhakrishna, A. Both, S. Shekarpour, I. Lytra, R. Usbeck, A. Vyas, A. Khikmatullaev, D. Punjani, C. Lange, et al., "Why Reinvent the Wheel: Let's Build Question Answering Systems Together," *Proceedings of the 2018 World Wide Web Conference on World Wide Web*, International World Wide Web Conferences Steering Committee, 2018 1247 (cit. on pp. 2, 3, 13, 39–41, 48).

[10] D. Diefenbach, V. Lopez, K. Singh, and P. Maret, *Core techniques of question answering systems over knowledge bases: a survey*, Knowledge and Information systems **55** (2018) 529 (cit. on pp. 2, 5, 14, 17, 39, 40, 62, 86).

[11] J.-D. Kim, C. Unger, A.-C. N. Ngomo, A. Freitas, Y.-g. Hahm, J. Kim, S. Nam, G.-H. Choi, J.-u. Kim, R. Usbeck, et al., *OKBQA Framework for collaboration on developing natural language question answering systems*, (2017), URL: http://sigir2017.okbqa.org/papers/OKBQA2017_paper_9.pdf (cit. on pp. 2, 3, 19, 39, 40).

[12] X. Zhang and L. Zou, "IMPROVE-QA: An Interactive Mechanism for RDF Question/Answering Systems," *Proceedings of the 2018 International Conference on Management of Data*, SIGMOD '18, ACM, 2018 1753 (cit. on pp. 3, 86).

[13] N. Jayaram, A. Khan, C. Li, X. Yan, and R. Elmasri, *Querying Knowledge Graphs by Example Entity Tuples*, IEEE Transactions on Knowledge and Data Engineering **27** (2015) 2797, ISSN: 1041-4347 (cit. on pp. 3, 86).

[14] K. Höffner, S. Walter, E. Marx, R. Usbeck, J. Lehmann, and A.-C. Ngonga Ngomo, *Survey on challenges of question answering in the semantic web*, Semantic Web **8** (2017) 895 (cit. on pp. 5, 17, 39, 40, 55, 86).

[15] C. Martinez-Cruz, I. J. Blanco, and M. A. Vila, *Ontologies versus relational databases: are they so different? A comparison*, Artificial Intelligence Review **38** (2012) 271 (cit. on p. 12).

[16] P. Ernst, C. Meng, A. Siu, and G. Weikum, "Knowlife: a knowledge graph for health and life sciences," *2014 IEEE 30th International Conference on Data Engineering* (cit. on p. 12).

[17] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor, "Freebase: a collaboratively created graph database for structuring human knowledge," *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, AcM, 2008 1247 (cit. on pp. 12, 17, 39, 42, 54).

[18] D. Vrandečić and M. Krötzsch, *Wikidata: a free collaborative knowledge base*, (2014) (cit. on p. 12).

[19] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. N. Mendes, S. Hellmann, M. Morsey, P. van Kleef, S. Auer, and C. Bizer, *DBpedia - A large-scale, multilingual knowledge base extracted from Wikipedia*, Semantic Web **6** (2015) 167 (cit. on p. 13).

[20] A. Abdelkawi, H. Zafar, M. Maleshkova, and J. Lehmann, "Complex Query Augmentation for Question Answering over Knowledge Graphs," *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*, 2019 (cit. on pp. 13, 18, 90).

[21] L. Hirschman and R. Gaizauskas, *Natural language question answering: the view from here*, natural language engineering **7** (2001) 275 (cit. on pp. 15, 16).

[22] O. Kolomiyets and M.-F. Moens, *A survey on question answering technology from an information retrieval perspective*, Information Sciences **181** (2011) 5412 (cit. on p. 16).

[23] M. Paşca, *Open-domain question answering from large text collections*, 2003 (cit. on p. 16).

[24] I. Androutsopoulos, G. D. Ritchie, and P. Thanisch,
Natural language interfaces to databases–an introduction,
Natural language engineering **1** (1995) 29 (cit. on p. 16).

[25] W. A. Woods, R. M. Kaplan, B. Nash-Webber, et al.,
The lunar sciences natural language information system: Final report,
BBN report **2378** (1972) (cit. on p. 16).

[26] B. F. Green Jr, A. K. Wolf, C. Chomsky, and K. Laughery,
"Baseball: an automatic question-answerer,"
Papers presented at the May 9-11, 1961, western joint IRE-AIEE-ACM computer conference,
ACM, 1961 219 (cit. on p. 16).

[27] I. Ritchie, P. Thanisch, et al., "MASQUE/sql-an efficient and portable natural language query interface for relational database," *Proc of Sixth International Conference on Industrial & Engineering Applications of Artificial Intelligence & Expert System*, 1993 (cit. on p. 16).

[28] R. D. Burke, K. J. Hammond, V. Kulyukin, S. L. Lytinen, N. Tomuro, and S. Schoenberg,
Question answering from frequently asked question files: Experiences with the faq finder system, AI magazine **18** (1997) 57 (cit. on p. 16).

[29] E. Riloff and M. Thelen,
"A rule-based question answering system for reading comprehension tests,"
Proceedings of the 2000 ANLP/NAACL Workshop on Reading comprehension tests as evaluation for computer-based language understanding sytems-Volume 6,
Association for Computational Linguistics, 2000 13 (cit. on p. 16).

[30] V. Zhong, C. Xiong, and R. Socher,
Seq2sql: Generating structured queries from natural language using reinforcement learning,
arXiv preprint arXiv:1709.00103 (2017) (cit. on pp. 16, 28).

[31] X. Xu, C. Liu, and D. Song,
Sqlnet: Generating structured queries from natural language without reinforcement learning,
arXiv preprint arXiv:1711.04436 (2017) (cit. on p. 16).

[32] T. Yu, R. Zhang, K. Yang, M. Yasunaga, D. Wang, Z. Li, J. Ma, I. Li, Q. Yao, S. Roman, et al.,
Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task, arXiv preprint arXiv:1809.08887 (2018) (cit. on p. 16).

[33] P. Yin, Z. Lu, H. Li, and B. Kao,
Neural enquirer: Learning to query tables with natural language,
arXiv preprint arXiv:1512.00965 (2015) (cit. on p. 16).

[34] V. Lopez, V. Uren, M. Sabou, and E. Motta,
Is question answering fit for the semantic web?: a survey, Semantic Web **2** (2011) 125
(cit. on pp. 16, 17).

[35] K. Affolter, K. Stockinger, and A. Bernstein,
A comparative survey of recent natural language interfaces for databases,
The VLDB Journal **28** (2019) 793 (cit. on p. 16).

[36] B. Katz, S. Felshin, D. Yuret, A. Ibrahim, J. Lin, G. Marton, A. J. McFarland, and B. Temelkuran, "Omnibase: Uniform access to heterogeneous data for question answering," *International Conference on Application of Natural Language to Information Systems*, Springer, 2002 230 (cit. on p. 16).

[37] E. Kaufmann and A. Bernstein, "How useful are natural language interfaces to the semantic web for casual end-users?" *The Semantic Web*, Springer, 2007 281 (cit. on p. 17).

[38] P. Cimiano, P. Haase, and J. Heizmann, "Porting natural language interfaces between domains: an experimental user study with the orakel system," *Proceedings of the 12th international conference on Intelligent user interfaces*, 2007 180 (cit. on p. 17).

[39] V. Lopez, V. Uren, E. Motta, and M. Pasin, *AquaLog: An ontology-driven question answering system for organizational semantic intranets*, Journal of Web Semantics **5** (2007) 72 (cit. on p. 17).

[40] S. Mithun, L. Kosseim, and V. Haarslev, "Resolving quantifier and number restriction to question owl ontologies," *Third International Conference on Semantics, Knowledge and Grid (SKG 2007)*, IEEE, 2007 218 (cit. on p. 17).

[41] O. Ferrández, R. Izquierdo, S. Ferrández, and J. L. Vicedo, *Addressing ontology-based question answering with collections of user queries*, Information Processing & Management **45** (2009) 175 (cit. on p. 17).

[42] D. Damljanovic, M. Agatonovic, and H. Cunningham, "Natural language interfaces to ontologies: Combining syntactic analysis and ontology-based lookup through the user interaction," *Extended Semantic Web Conference*, Springer, 2010 106 (cit. on p. 17).

[43] S. Linckels and C. Meinel, "A simple solution for an intelligent librarian system.," *IADIS AC*, 2005 495 (cit. on p. 17).

[44] P. Cimiano and M. Minock, "Natural language interfaces: what is the problem?–a data-driven quantitative analysis," *International Conference on Application of Natural Language to Information Systems*, Springer, 2009 192 (cit. on p. 17).

[45] S. J. Athenikos and H. Han, *Biomedical question answering: A survey*, Computer methods and programs in biomedicine **99** (2010) 1 (cit. on p. 17).

[46] V. Lopez, C. Unger, P. Cimiano, and E. Motta, *Evaluating question answering over linked data*, Journal of Web Semantics **21** (2013) 3 (cit. on p. 17).

[47] A. Freitas, E. Curry, J. G. Oliveira, and S. O'Riain, *Querying heterogeneous datasets on the linked data web: challenges, approaches, and trends*, IEEE Internet Computing **16** (2011) 24 (cit. on p. 17).

[48] M. Dubey, S. Dasgupta, A. Sharma, K. Höffner, and J. Lehmann, "Asknow: A framework for natural language query formalization in sparql," *International Semantic Web Conference*, Springer, 2016 300 (cit. on pp. 17, 25, 27, 33, 40, 44, 55, 86, 89).

[49] C. Unger, L. Bühmann, J. Lehmann, A.-C. Ngonga Ngomo, D. Gerber, and P. Cimiano, "Template-based question answering over RDF data," *Proceedings of the 21st international conference on World Wide Web*, ACM, 2012 639 (cit. on pp. 17, 27, 55, 56, 89).

[50] Q. Cai and A. Yates, "Large-scale semantic parsing via schema matching and lexicon extension," *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, 2013 (cit. on pp. 17, 18, 89, 91, 92).

[51] A. Bordes, N. Usunier, S. Chopra, and J. Weston, *Large-scale simple question answering with memory networks*, arXiv preprint arXiv:1506.02075 (2015) (cit. on pp. 18, 54, 86, 89, 92).

[52] J. Berant, A. Chou, R. Frostig, and P. Liang, "Semantic parsing on freebase from question-answer pairs," *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, 2013 1533 (cit. on pp. 18, 41, 54, 89, 90, 92).

[53] I. V. Serban, A. García-Durán, C. Gulcehre, S. Ahn, S. Chandar, A. Courville, and Y. Bengio, *Generating factoid questions with recurrent neural networks: The 30m factoid question-answer corpus*, arXiv preprint arXiv:1603.06807 (2016) (cit. on pp. 18, 89, 91, 92).

[54] P. Trivedi, G. Maheshwari, M. Dubey, and J. Lehmann, "Lc-quad: A corpus for complex question answering over knowledge graphs," *International Semantic Web Conference*, Springer, 2017 210 (cit. on pp. 18, 60, 90–92).

[55] M. Dubey, D. Banerjee, A. Abdelkawi, and J. Lehmann, "Lc-quad 2.0: A large dataset for complex question answering over wikidata and dbpedia," *International Semantic Web Conference*, Springer, 2019 69 (cit. on pp. 18, 90–92).

[56] N. Chakraborty, D. Lukovnikov, G. Maheshwari, P. Trivedi, J. Lehmann, and A. Fischer, *Introduction to Neural Network based Approaches for Question Answering over Knowledge Graphs*, (2019) (cit. on pp. 18, 55).

[57] W.-t. Yih, M. Richardson, C. Meek, and M.-W. Chang, "The Value of Semantic Parse Labeling for Knowledge Base Question Answering.," *54th Annual Meeting of the Association for Computational Linguistics*, 2016 201 (cit. on pp. 18, 92).

[58] J. Bao, N. Duan, Z. Yan, M. Zhou, and T. Zhao, "Constraint-based question answering with knowledge graph," *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, 2016 2503 (cit. on pp. 18, 92).

[59] Y. Su, H. Sun, B. Sadler, M. Srivatsa, I. Gur, Z. Yan, and X. Yan, "On generating characteristic-rich question sets for qa evaluation," *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 2016 (cit. on pp. 18, 92).

[60] A. Talmor and J. Berant, *The web as a knowledge-base for answering complex questions*, arXiv preprint arXiv:1803.06643 (2018) (cit. on pp. 18, 92).

[61] A. Abujabal, R. S. Roy, M. Yahya, and G. Weikum, *ComQA: A Community-sourced Dataset for Complex Factoid Question Answering with Paraphrase Clusters*, arXiv preprint arXiv:1809.09528 (2018) (cit. on pp. 18, 92).

[62] M. Azmy, P. Shi, J. Lin, and I. Ilyas, "Farewell freebase: Migrating the simplequestions dataset to dbpedia," *Proceedings of the 27th International Conference on Computational Linguistics*, 2018 2093 (cit. on pp. 18, 92).

[63] A. Saha, V. Pahuja, M. M. Khapra, K. Sankaranarayanan, and S. Chandar, "Complex sequential question answering: Towards learning to converse over linked question answer pairs with a knowledge graph," *Thirty-Second AAAI Conference*, 2018 (cit. on pp. 18, 91, 92).

[64] K. Jiang, D. Wu, and H. Jiang, "FreebaseQA: A New Factoid QA Data Set Matching Trivia-Style Question-Answer Pairs with Freebase," *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics* (cit. on pp. 18, 91, 92).

[65] E. Marx, R. Usbeck, A.-C. N. Ngomo, K. Höffner, J. Lehmann, and S. Auer, "Towards an open question answering architecture," *Proceedings of the 10th International Conference on Semantic Systems*, ACM, 2014 57 (cit. on p. 19).

[66] A. Both, D. Diefenbach, K. Singh, S. Shekarpour, D. Cherix, and C. Lange, "Qanary–a methodology for vocabulary-driven open question answering systems," *International Semantic Web Conference*, Springer, 2016 625 (cit. on p. 19).

[67] K. Singh, I. Lytra, M.-E. Vidal, D. Punjani, H. Thakkar, C. Lange, and S. Auer, "QA estro–semantic-based composition of question answering pipelines," *International Conference on Database and Expert Systems Applications*, Springer, 2017 19 (cit. on p. 19).

[68] Ó. Ferrández, C. Spurk, M. Kouylekov, I. Dornescu, S. Ferrández, M. Negri, R. Izquierdo, D. Tomás, C. Orasan, G. Neumann, et al., *The QALL-ME framework: A specifiable-domain multilingual question answering architecture*, Journal of Web Semantics **9** (2011) 137 (cit. on p. 19).

[69] A. Sakor, I. Mulang, K. Singh, A. Shekarpour, M. Vidal, J. Lehmann, and S. Auer, "Old is Gold: Linguistic Driven Approach for Entity and Relation Linking of Short Text," *Proceedings of the 2019 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, 2019 (cit. on pp. 25, 27, 33, 35).

[70] M. Dubey, D. Banerjee, D. Chaudhuri, and J. Lehmann,
EARL: Joint Entity and Relation Linking for Question Answering over Knowledge Graphs,
version 1, (2018), arXiv: http://arxiv.org/abs/1801.03825v1 [cs.AI, cs.CL],
URL: http://arxiv.org/abs/1801.03825v1 (cit. on pp. 25, 27, 49, 72, 86, 101).

[71] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa,
Natural language processing (almost) from scratch,
Journal of Machine Learning Research **12** (2011) 2493 (cit. on pp. 25, 72).

[72] A. Akbik, D. Blythe, and R. Vollgraf,
"Contextual String Embeddings for Sequence Labeling,"
COLING 2018, 27th International Conference on Computational Linguistics, 2018
(cit. on pp. 25, 33).

[73] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and
M. Riedmiller, *Playing atari with deep reinforcement learning*,
arXiv preprint arXiv:1312.5602 (2013) (cit. on p. 26).

[74] M. Tavakol and U. Brefeld, "Factored MDPs for detecting topics of user sessions,"
Proceedings of the 8th ACM Conference on Recommender Systems, ACM, 2014 33
(cit. on p. 26).

[75] G. H. Lee and K. J. Lee,
"Automatic Text Summarization Using Reinforcement Learning with Embedding Features,"
International Joint Conference on Natural Language Processing, 2017 (cit. on p. 26).

[76] D. Yogatama, P. Blunsom, C. Dyer, E. Grefenstette, and W. Ling,
"Learning to compose words into sentences with reinforcement learning,"
5th International Conference on Learning Representations, 2017 (cit. on p. 26).

[77] S. P. Singh, M. J. Kearns, D. J. Litman, and M. A. Walker,
"Reinforcement learning for spoken dialogue systems,"
Advances in Neural Information Processing Systems, 2000 956 (cit. on p. 26).

[78] J. Li, W. Monroe, A. Ritter, D. Jurafsky, M. Galley, and J. Gao,
"Deep Reinforcement Learning for Dialogue Generation,"
Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing,
2016 1192 (cit. on p. 26).

[79] P. Ferragina and U. Scaiella, *Fast and accurate annotation of short texts with wikipedia pages*,
IEEE software **29** (2012) 70 (cit. on pp. 27, 33, 49).

[80] J. Hoffart, M. A. Yosef, I. Bordino, H. Fürstenau, M. Pinkal, M. Spaniol, B. Taneva, S. Thater,
and G. Weikum, "Robust disambiguation of named entities in text,"
Proceedings of the Conference on Empirical Methods in Natural Language Processing, 2011
(cit. on p. 27).

[81] P. N. Mendes, M. Jakob, A. García-Silva, and C. Bizer,
"DBpedia spotlight: shedding light on the web of documents,"
Proceedings of the 7th international conference on semantic systems, 2011
(cit. on pp. 27, 33).

[82] A. Moro, A. Raganato, and R. Navigli,
Entity linking meets word sense disambiguation: a unified approach,
Transactions of the Association for Computational Linguistics (2014) (cit. on pp. 27, 33).

[83] R. Speck and A.-C. N. Ngomo, "Ensemble learning for named entity recognition,"
International semantic web conference, 2014 (cit. on pp. 27, 33).

[84] Y. Yang and M.-W. Chang,
S-mart: Novel tree-based structured learning algorithms applied to tweet entity linking,
(2016) (cit. on p. 27).

[85] W. Shen, J. Wang, and J. Han,
Entity linking with a knowledge base: Issues, techniques, and solutions,
IEEE Transactions on Knowledge and Data Engineering **27** (2015) 443 (cit. on p. 27).

[86] I. O. Mulang, K. Singh, and F. Orlandi,
"Matching natural language relations to knowledge graph properties for question answering,"
Proceedings of the 13th International Conference on Semantic Systems, ACM, 2017 89
(cit. on pp. 27, 33).

[87] D. Gerber and A.-C. N. Ngomo, "Bootstrapping the linked data web,"
1st Workshop on Web Scale Knowledge Extraction@ ISWC'11 (cit. on p. 27).

[88] N. Nakashole, G. Weikum, and F. Suchanek,
"PATTY: a taxonomy of relational patterns with semantic types,"
Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (cit. on p. 27).

[89] D. Zelenko, C. Aone, and A. Richardella, *Kernel methods for relation extraction*,
Journal of machine learning research (2003) (cit. on p. 27).

[90] M. Mintz, S. Bills, R. Snow, and D. Jurafsky,
"Distant supervision for relation extraction without labeled data,"
Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL, 2009
(cit. on p. 27).

[91] K. Fundel, R. Küffner, and R. Zimmer,
RelEx-Relation extraction using dependency parse trees, Bioinformatics (2006) (cit. on p. 27).

[92] J. Lehmann, T. Furche, G. Grasso, A.-C. N. Ngomo, C. Schallhart, A. Sellers, C. Unger,
L. Bühmann, D. Gerber, K. Höffner, et al.,
"Deqa: deep web extraction for question answering," *International Semantic Web Conference*,
Springer, 2012 131 (cit. on p. 27).

[93] L. Zou, R. Huang, H. Wang, J. X. Yu, W. He, and D. Zhao,
"Natural Language Question Answering over RDF: A Graph Data Driven Approach,"
Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data
(cit. on p. 27).

[94] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa,
Natural Language Processing (Almost) from Scratch, J. Mach. Learn. Res. **12** (2011) 2493,
ISSN: 1532-4435, URL: http://dl.acm.org/citation.cfm?id=1953048.2078186
(cit. on pp. 27, 33).

[95] P. Trivedi and M. Dubey,
"A Corpus for Complex Question Answering over Knowledge Graphs,"
16th International Semantic Web Conference, 2017 (cit. on pp. 27, 32, 48).

[96] P. Shah, D. Hakkani-Tür, G. Tür, A. Rastogi, A. Bapna, N. Nayak, and L. Heck,
Building a conversational agent overnight with dialogue self-play,
arXiv preprint arXiv:1801.04871 (2018) (cit. on p. 28).

[97] P.-S. Huang, C. Wang, R. Singh, W.-t. Yih, and X. He,
Natural language to structured query generation via meta-learning,
arXiv preprint arXiv:1803.02400 (2018) (cit. on p. 28).

[98] J. Andreas, M. Rohrbach, T. Darrell, and D. Klein,
Learning to compose neural networks for question answering,
arXiv preprint arXiv:1601.01705 (2016) (cit. on p. 28).

[99] R. Takanobu, T. Zhang, J. Liu, and M. Huang,
"A hierarchical framework for relation extraction with reinforcement learning,"
Proceedings of the AAAI Conference on AI, 2019 (cit. on p. 28).

[100] R. S. Sutton and A. G. Barto, *Introduction to reinforcement learning*, vol. 135,
MIT press Cambridge, 1998 (cit. on p. 28).

[101] R. S. Sutton, D. A. McAllester, S. Singh, and Y. Mansour,
"Policy gradient methods for reinforcement learning with function approximation,"
Advances in neural information processing systems, 2000 (cit. on p. 31).

[102] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation,"
Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), 2014 1532 (cit. on pp. 31, 32, 46, 61).

[103] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean,
"Distributed representations of words and phrases and their compositionality,"
Advances in neural information processing systems, 2013 (cit. on p. 31).

[104] C. Unger, A.-C. N. Ngomo, and E. Cabrio,
"6th open challenge on question answering over linked data (qald-6),"
Semantic Web Evaluation Challenge, Springer, 2016 171 (cit. on p. 32).

[105] R. Usbeck, A.-C. N. Ngomo, B. Haarmann, A. Krithara, M. Röder, and G. Napolitano,
"7th open challenge on question answering over linked data (QALD-7),"
Semantic Web Evaluation Challenge, Springer, 2017 59 (cit. on p. 32).

[106] R. Martinez-Cantin, K. Tee, and M. McCourt,
"Practical Bayesian optimization in the presence of outliers,"
Proceedings of the 21th International Conference on Artificial Intelligence and Statistics,
2018 (cit. on p. 32).

[107] P. Christen, "A comparison of personal name matching: Techniques and practical issues,"
Sixth IEEE International Conference on Data Mining-Workshops (ICDMW'06), IEEE, 2006
290 (cit. on pp. 32, 69).

[108] M. Kusner, Y. Sun, N. Kolkin, and K. Weinberger,
"From word embeddings to document distances,"
International Conference on Machine Learning, 2015 957 (cit. on pp. 32, 58).

[109] K. Singh, I. Mulang, I. Lytra, M. Jaradeh, A. Sakor, M. Vidal, C. Lange, and S. Auer,
"Capturing knowledge in semantically-typed relational patterns to enhance relation linking,"
Proceedings of the Knowledge Capture Conference, ACM, 2017 31 (cit. on p. 33).

[110] H. Zafar, G. Napolitano, and J. Lehmann,
"Formal query generation for question answering over knowledge bases,"
European Semantic Web Conference, Springer, 2018 714
(cit. on pp. 35, 36, 54, 56, 57, 62–64, 73, 86).

[111] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives,
Dbpedia: A nucleus for a web of open data, The semantic web (2007) 722
(cit. on pp. 39, 40, 48, 59).

[112] H. Bast and E. Haussmann, "More accurate question answering on freebase," *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, ACM, 2015 1431 (cit. on pp. 40, 41).

[113] A. Abujabal, M. Yahya, M. Riedewald, and G. Weikum,
"Automated template generation for question answering over knowledge graphs,"
Proceedings of the 26th international conference on world wide web,
International World Wide Web Conferences Steering Committee, 2017 1191
(cit. on pp. 40, 41, 56).

[114] S. He, Y. Zhang, K. Liu, and J. Zhao,
"CASIA@ V2: A MLN-based Question Answering System over Linked Data.," 2014
(cit. on p. 40).

[115] S. Shekarpour, E. Marx, A.-C. N. Ngomo, and S. Auer,
Sina: Semantic interpretation of user queries for question answering on interlinked data,
Web Semantics: Science, Services and Agents on the World Wide Web **30** (2015) 39
(cit. on pp. 40, 55).

[116] D. Lukovnikov, A. Fischer, J. Lehmann, and S. Auer, "Neural Network-based Question Answering over Knowledge Graphs on Word and Character Level,"
Proceedings of the 26th International Conference on World Wide Web,
International World Wide Web Conferences Steering Committee, 2017 1211
(cit. on pp. 40, 86).

[117] A. Fader, L. Zettlemoyer, and O. Etzioni,
"Paraphrase-driven learning for open question answering,"
Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics,
vol. 1, 2013 1608 (cit. on p. 41).

[118] S. W.-t. Yih, M.-W. Chang, X. He, and J. Gao, *Semantic parsing via staged query graph generation: Question answering with knowledge base*, (2015) (cit. on pp. 44, 45, 54).

[119] V. Lopez, M. Fernández, E. Motta, and N. Stieler,
Poweraqua: Supporting users in querying and exploring the semantic web,
Semantic Web (2012) 249 (cit. on p. 44).

[120] K. S. Tai, R. Socher, and C. D. Manning, "Improved Semantic Representations From Tree-Structured Long Short-Term Memory Networks," *ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, 2015 1556 (cit. on pp. 45, 46, 50).

[121] Y. Bengio, P. Simard, and P. Frasconi, *Learning long-term dependencies with gradient descent is difficult*, IEEE transactions on neural networks (1994) (cit. on p. 46).

[122] S. Hochreiter and J. Schmidhuber, *Long short-term memory*, Neural computation **9** (1997) 1735 (cit. on p. 46).

[123] D. Chen and C. Manning, "A fast and accurate dependency parser using neural networks," *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014 740 (cit. on p. 50).

[124] J. Mueller and A. Thyagarajan, "Siamese Recurrent Architectures for Learning Sentence Similarity.," 2016 (cit. on p. 51).

[125] J. Berant and P. Liang, "Semantic parsing via paraphrasing," *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2014 1415 (cit. on p. 54).

[126] W. Yin, M. Yu, B. Xiang, B. Zhou, and H. Schütze, *Simple question answering by attentive convolutional neural network*, arXiv preprint arXiv:1606.03391 (2016) (cit. on p. 54).

[127] A. Bordes, S. Chopra, and J. Weston, *Question answering with subgraph embeddings*, arXiv preprint arXiv:1406.3676 (2014) (cit. on p. 54).

[128] D. Diefenbach, A. Both, K. Singh, and P. Maret, *Towards a question answering system over the Semantic Web*, Semantic Web (2018) 1 (cit. on p. 54).

[129] G. Maheshwari, P. Trivedi, D. Lukovnikov, N. Chakraborty, A. Fischer, and J. Lehmann, "Learning to Rank Query Graphs for Complex Question Answering over Knowledge Graphs," *International Semantic Web Conference*, Springer, 2019 (cit. on p. 54).

[130] S. Walter, C. Unger, P. Cimiano, and D. Bär, "Evaluation of a layered approach to question answering over linked data," *International Semantic Web Conference*, Springer, 2012 362 (cit. on p. 55).

[131] H. SZ et al., *CASIA@ V2: A MLN-based question answering system over linked data*, (2014) (cit. on p. 55).

[132] S. Hakimov, C. Unger, S. Walter, and P. Cimiano, "Applying semantic parsing to question answering over linked data: Addressing the lexical gap," *International Conference on Applications of Natural Language to Information Systems*, Springer, 2015 103 (cit. on p. 55).

[133] L. S. Zettlemoyer and M. Collins, *Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars*, arXiv preprint arXiv:1207.1420 (2012) (cit. on p. 55).

[134] T. Hamon, N. Grabar, F. Mougin, and F. Thiessard,
"Description of the POMELO System for the Task 2 of QALD-2014.,"
CLEF (Working Notes), 2014 1212 (cit. on p. 55).

[135] N. Ngomo, *9th Challenge on Question Answering over Linked Data (QALD-9)*,
language **7** () 1 (cit. on p. 60).

[136] P. Trivedi, G. Maheshwari, M. Dubey, and J. Lehmann,
"LC-QuAD: A Corpus for Complex Question Answering over Knowledge Graphs,"
The Semantic Web – ISWC 2017, Springer International Publishing, 2017 210
(cit. on pp. 65, 74, 86).

[137] E. Demidova, X. Zhou, and W. Nejdl, "Efficient Query Construction for Large Scale Data,"
Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '13, ACM, 2013 573 (cit. on pp. 66, 70, 85).

[138] S. Lomax and S. Vadera, *A survey of cost-sensitive decision tree induction algorithms*,
ACM Computing Surveys (CSUR) **45** (2013) 16 (cit. on p. 68).

[139] D. Diefenbach, A. Both, K. D. Singh, and P. Maret,
Towards a Question Answering System over the Semantic Web,
Semantic Web **Pre-press** (2019) (cit. on pp. 72, 75, 77, 79, 80, 86).

[140] H. Zafar, M. Tavakol, and J. Lehmann,
MDP-based Shallow Parsing in Distantly Supervised QA Systems, (2019) (cit. on p. 72).

[141] S. Bird, E. Klein, and E. Loper,
Natural language processing with Python: analyzing text with the natural language toolkit,
" O'Reilly Media, Inc.", 2009 (cit. on p. 72).

[142] F. Hasibi, K. Balog, and S. E. Bratsberg,
"On the reproducibility of the TAGME Entity Linking System,"
Proceedings of 38th European Conference on Information Retrieval, ECIR '16,
Springer, 2016 436 (cit. on pp. 72, 86).

[143] A.-C. Ngonga Ngomo, L. Bühmann, C. Unger, J. Lehmann, and D. Gerber,
"SPARQL2NL: verbalizing sparql queries,"
Proceedings of the 22nd International Conference on World Wide Web, ACM, 2013 329
(cit. on pp. 73, 90, 99).

[144] R. Usbeck, M. Röder, M. Hoffmann, F. Conrads, J. Huthmann, A.-C. Ngonga-Ngomo,
C. Demmler, and C. Unger, *Benchmarking question answering systems*,
Semantic Web (2019) 1 (cit. on p. 77).

[145] E. Demidova, X. Zhou, and W. Nejdl,
A Probabilistic Scheme for Keyword-Based Incremental Query Construction,
IEEE Trans. on Knowl. and Data Eng. **24** (2012) 426, ISSN: 1041-4347 (cit. on p. 85).

[146] E. Demidova, X. Zhou, and W. Nejdl, "FreeQ: an interactive query interface for freebase,"
Proceedings of the 21st World Wide Web Conference, WWW 2012, Lyon, France, April 16-20, 2012 (Companion Volume), 2012 325 (cit. on p. 85).

[147] E. Demidova, I. Oelze, and W. Nejdl, "Aligning freebase with the YAGO ontology," *Proceedings of the 22nd ACM International Conference on Information and Knowledge Management, CIKM'13*, 2013 579 (cit. on p. 85).

[148] S. W.-t. Yih, M.-W. Chang, X. He, and J. Gao, "Semantic Parsing via Staged Query Graph Generation: Question Answering with Knowledge Base," ACL the Association for Computational Linguistics, 2015 (cit. on p. 86).

[149] D. Mottin, M. Lissandrini, Y. Velegrakis, and T. Palpanas, *Exemplar Queries: Give Me an Example of What You Need*, Proc. VLDB Endow. **7** (2014) 365, ISSN: 2150-8097 (cit. on p. 86).

[150] Y. Su, S. Yang, H. Sun, M. Srivatsa, S. Kase, M. Vanni, and X. Yan, "Exploiting Relevance Feedback in Knowledge Graph Search," *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, ACM, 2015 1135, ISBN: 978-1-4503-3664-2 (cit. on p. 86).

[151] W. Zheng, H. Cheng, L. Zou, J. X. Yu, and K. Zhao, "Natural Language Question/Answering: Let Users Talk With The Knowledge Graph," *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, CIKM '17, ACM, 2017 217, ISBN: 978-1-4503-4918-5 (cit. on p. 86).

[152] S. Ferré, *Sparklis: An expressive query builder for SPARQL endpoints with guidance in natural language*, Semantic Web **8** (2017) 405 (cit. on p. 86).

[153] S. Gottschalk and E. Demidova, "EventKG+TL: Creating Cross-Lingual Timelines from an Event-Centric Knowledge Graph," *The Semantic Web: ESWC 2018 Satellite Events*, vol. 11155, Lecture Notes in Computer Science, Springer, 2018 164 (cit. on p. 86).

[154] L. Bradesko, M. J. Witbrock, J. Starc, Z. Herga, M. Grobelnik, and D. Mladenic, *Curious Cat-Mobile, Context-Aware Conversational Crowdsourcing Knowledge Acquisition*, ACM Trans. Inf. Syst. **35** (2017) 33:1 (cit. on p. 86).

[155] F. Li and H. V. Jagadish, *Constructing an Interactive Natural Language Interface for Relational Databases*, Proc. VLDB Endow. **8** (2014) 73, ISSN: 2150-8097 (cit. on pp. 86, 87).

[156] Y. Su, A. H. Awadallah, M. Wang, and R. W. White, "Natural Language Interfaces with Fine-Grained User Interaction: A Case Study on Web APIs," *Proceedings of the SIGIR 2018*, ACM, 2018 (cit. on p. 86).

[157] Z. Yao, X. Li, J. Gao, B. Sadler, and H. Sun, "Interactive semantic parsing for if-then recipes via hierarchical reinforcement learning," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019 2547 (cit. on p. 87).

[158] I. Gur, S. Yavuz, Y. Su, and X. Yan, "DialSQL: Dialogue Based Structured Query Generation," *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018*, Association for Computational Linguistics, 2018 (cit. on p. 87).

[159] Z. Yao, Y. Su, H. Sun, and W.-t. Yih, *Model-based Interactive Semantic Parsing: A Unified Framework and A Text-to-SQL Case Study*,
(2019) 5446, ed. by K. Inui, J. Jiang, V. Ng, and X. Wan (cit. on p. 87).

[160] S. Ferré, *Sparklis: an expressive query builder for SPARQL endpoints with guidance in natural language*, Semantic Web **8** (2017) 405 (cit. on p. 90).

[161] W. Zheng, H. Cheng, L. Zou, J. X. Yu, and K. Zhao,
"Natural language question/answering: Let users talk with the knowledge graph,"
Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, 2017 (cit. on pp. 90, 99).

[162] B. Ell, A. Harth, and E. Simperl,
"Sparql query verbalization for explaining semantic search engine queries,"
European Semantic Web Conference, Springer, 2014 426 (cit. on pp. 90, 99).

[163] D. Diefenbach, Y. Dridi, K. Singh, and P. Maret,
"SPARQLtoUser: Did the question answering system understand me?" 2017
(cit. on pp. 90, 92, 99).

[164] M. Eric and C. D. Manning, *Key-value retrieval networks for task-oriented dialogue*,
arXiv preprint arXiv:1705.05414 (2017) (cit. on p. 92).

[165] D. Bahdanau, K. Cho, and Y. Bengio,
Neural Machine Translation by Jointly Learning to Align and Translate,
arXiv e-prints, arXiv:1409.0473 (2014) arXiv:1409.0473, arXiv: 1409.0473 [cs.CL]
(cit. on pp. 101, 103).

[166] M.-T. Luong, H. Pham, and C. D. Manning,
Effective Approaches to Attention-based Neural Machine Translation,
arXiv e-prints, arXiv:1508.04025 (2015) arXiv:1508.04025, arXiv: 1508.04025 [cs.CL]
(cit. on pp. 101, 103).

[167] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, *Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation*, arXiv e-prints, arXiv:1406.1078 (2014) arXiv:1406.1078,
arXiv: 1406.1078 [cs.CL] (cit. on p. 101).

[168] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin,
Convolutional Sequence to Sequence Learning,
arXiv e-prints, arXiv:1705.03122 (2017) arXiv:1705.03122, arXiv: 1705.03122 [cs.CL]
(cit. on pp. 101, 103).

[169] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, *Attention Is All You Need*,
arXiv e-prints, arXiv:1706.03762 () arXiv:1706.03762, arXiv: 1706.03762 [cs.CL]
(cit. on pp. 101, 103).

[170] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu,
"Bleu: a Method for Automatic Evaluation of Machine Translation,"
Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, 2002 (cit. on p. 101).

[171] T. Luong, M. Kayser, and C. D. Manning,
"Deep Neural Language Models for Machine Translation,"
Proceedings of the 19th Conference on Computational Natural Language Learning,
Association for Computational Linguistics, 2015 (cit. on p. 102).

[172] C. Longoni, A. Bonezzi, and C. K. Morewedge, *Resistance to Medical Artificial Intelligence*,
Journal of Consumer Research **46** (2019) 629, ISSN: 0093-5301, eprint:
https://academic.oup.com/jcr/article-pdf/46/4/629/30624402/ucz013.pdf,
URL: https://doi.org/10.1093/jcr/ucz013 (cit. on p. 108).

[173] K. Xu, S. Zhang, Y. Feng, and D. Zhao,
"Answering natural language questions via phrasal semantic parsing,"
Natural Language Processing and Chinese Computing, Springer, 2014 333.

[174] E. Cabrio, J. Cojan, A. P. Aprosio, B. Magnini, A. Lavelli, and F. Gandon,
"QAKiS: an open domain QA system based on relational patterns," *Proceedings of the 2012th International Conference on Posters & Demonstrations Track-Volume 914*,
CEUR-WS. org, 2012 9.

[175] D. A. Lindberg, B. L. Humphreys, and A. T. McCray, *The unified medical language system*,
Yearbook of Medical Informatics **2** (1993) 41.

[176] M. Wick, *GeoNames*, GeoNames, 2006.

[177] G. A. Miller, *WordNet: a lexical database for English*,
Communications of the ACM **38** (1995) 39.

[178] M. Dubey, D. Banerjee, A. Abdelkawi, and J. Lehmann, "LC-QuAD 2.0: A Large Dataset for Complex Question Answering over Wikidata and DBpedia,"
Proceedings of the 18th International Semantic Web Conference (ISWC), Springer, 2019.

[179] D. Diefenbach, S. Amjad, A. Both, K. D. Singh, and P. Maret,
"Trill: A Reusable Front-End for QA Systems," *ESWC 2017 Satellite Events*, 2017 48.

[180] W. Zheng, H. Cheng, L. Zou, J. X. Yu, and K. Zhao,
"Natural Language Question/Answering: Let Users Talk With The Knowledge Graph,"
Proc. of the ACM CIKM 2017, 2017 217.

[181] M. Dubey, D. Banerjee, D. Chaudhuri, and J. Lehmann,
"EARL: Joint Entity and Relation Linking for Question Answering over Knowledge Graphs,"
Proceedings of the ISWC 2018 - 17th International Semantic Web Conference, Part I, 2018 108.

[182] S. Liu, S. S. Bhowmick, W. Zhang, S. Wang, and W. Huang, *NEURON: An Interactive Natural Language Interface for Understanding Query Execution Plans in RDBMS*,
arXiv preprint arXiv:1805.05670 (2018).

[183] A.-C. Ngonga Ngomo, L. Bühmann, C. Unger, J. Lehmann, and D. Gerber,
"Sorry, i don't speak SPARQL: translating SPARQL queries into natural language,"
Proceedings of the 22nd international conference on World Wide Web, ACM, 2013 977.

[184] C. Unger, C. Forascu, V. Lopez, A.-C. N. Ngomo, E. Cabrio, P. Cimiano, and S. Walter,
"Question answering over linked data (QALD-4)," *Working Notes for CLEF 2014 Conference*,
2014.

[185] D. Guo, D. Tang, N. Duan, M. Zhou, and J. Yin,
"Dialog-to-action: conversational question answering over a large-scale knowledge base,"
NeurIPS, 2018.

[186] P. Cimiano, V. Lopez, C. Unger, E. Cabrio, A.-C. N. Ngomo, and S. Walter,
"Multilingual question answering over linked data (qald-3): Lab overview,"
International conference of the cross-language evaluation forum for european languages,
Springer, 2013 321.

[187] R. Usbeck, A.-C. N. Ngomo, F. Conrads, M. Röder, and G. Napolitano,
8th Challenge on Question Answering over Linked Data (QALD-8), language **7** (2018) 1.

[188] S. Vakulenko, J. D. Fernandez Garcia, A. Polleres, M. de Rijke, and M. Cochez,
"Message Passing for Complex Question Answering over Knowledge Graphs,"
Proceedings of 28th International Conference on Information and Knowledge Management.

[189] Y. Lan, S. Wang, and J. Jiang, "Knowledge base question answering with topic units,"
Proceedings of the 28th International Joint Conference on Artificial Intelligence.

[190] C. Liang, J. Berant, Q. Le, K. D. Forbus, and N. Lao,
Neural symbolic machines: Learning semantic parsers on freebase with weak supervision,
arXiv preprint arXiv:1611.00020 (2016).

[191] Y. Chen, L. Wu, and M. J. Zaki,
Bidirectional Attentive Memory Networks for Question Answering over Knowledge Bases,
arXiv preprint arXiv:1903.02188 (2019).

[192] J. Cheng, S. Reddy, V. Saraswat, and M. Lapata,
Learning structured natural language representations for semantic parsing,
arXiv preprint arXiv:1704.08387 (2017).

[193] K. Luo, F. Lin, X. Luo, and K. Zhu,
"Knowledge base question answering via encoding of complex query graphs,"
Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing,
2018 2185.

[194] S. Shin, X. Jin, J. Jung, and K.-H. Lee,
Predicate constraints based question answering over knowledge graph,
Information Processing & Management (2019).

[195] V. Uren, M. Sabou, E. Motta, M. Fernandez, V. Lopez, and Y. Lei,
Reflections on five years of evaluating semantic search systems,
International Journal of Metadata, Semantics and Ontologies **5** (2010) 87.

[196] A. Culotta and J. Sorensen, "Dependency tree kernels for relation extraction,"
Proceedings of the 42nd annual meeting on association for computational linguistics, 2004.

[197] D. Lukovnikov, A. Fischer, S. Auer, and J. Lehmann, "Neural Network-based Question Answering over Knowledge Graphs on Word and Character Level,"
Proceedings of the 26th international conference on World Wide Web, 2017,
URL: http://jens-lehmann.org/files/2017/www_nn_factoid_qa.pdf.

[198] J. Ebrahimi, A. Rao, D. Lowd, and D. Dou,
Hotflip: White-box adversarial examples for text classification,
arXiv preprint arXiv:1712.06751 (2017).

[199] P.-S. Huang, R. Stanforth, J. Welbl, C. Dyer, D. Yogatama, S. Gowal, K. Dvijotham, and P. Kohli, *Achieving verified robustness to symbol substitutions via interval bound propagation*, arXiv preprint arXiv:1909.01492 (2019).

[200] C. Dyer, M. Ballesteros, W. Ling, A. Matthews, and N. A. Smith,
"Transition-Based Dependency Parsing with Stack Long Short-Term Memory,"
Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics,
2015.

List of Figures

List of Tables