Towards Complex Question Answering over Knowledge Graphs

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

von
Mohnish Dubey
aus
Pipariya, Madhya Pradesh, India

Bonn, 21.07.2020

Angefertigt mit Genehmigung der Mathematisch-Naturwissenschaftlichen Fakultät der Rheinischen Friedrich-Wilhelms-Universität Bonn

1. Gutachter:	Prof. Dr. Jens Lehmann
2. Gutachter:	Prof. Dr. Sören Auer
Tag der Promotion:	19.01.2021
Erscheinungsjahr:	2021

Abstract

Over the past decade, Knowledge Graphs (KG) have emerged as a prominent repository for storing facts about the world in a linked data architecture. Providing machines with the capability of exploring such Knowledge Graphs and answering natural language questions over them, has been an active area of research. The purpose of this work, is to delve further into the research of retrieving information stored in KGs, based on the natural language questions posed by the user. Knowledge Graph Question Answering (KGQA) aims to produce a concise answer to a user question, such that the user is exempt from using KG vocabulary and overheads of learning a formal query language. Existing KGQA systems have achieved excellent results over Simple Questions, where the information required is limited to a single triple and a single formal query pattern. Our motivation is to improve the performance of KGQA over Complex Questions, where formal query patterns significantly vary, and a single triple is not confining for all the required information. Complex KGQA provides several challenges such as understanding semantics and syntactic structure of questions, Entity Linking, Relation Linking and Answer Representation. Lack of suitable datasets for complex question answering further adds to research gaps. Hence, in this thesis, we focus the research objective of laying the foundations for the advancement of the state-of-the-art for Complex Question Answering over Knowledge Graphs, by providing techniques to solve various challenges and provide resources to fill the research gaps.

First, we propose Normalized Query Structure (*NQS*), which is a linguistic analyzer module that helps the QA system to detect inputs and intents and the relation between them in the users' question. NQS acts like an intermediate language between natural language questions and formal expressions to ease the process of query formulation for complex questions. We then developed a framework named *LC-QuAD* to generate large scale question answering dataset by reversing the process of question answering, thereby translating natural language questions from the formal query using intermediate templates. Our goal is to use this framework for high variations in the query patterns and create a large size dataset with minimum human effort. The first version of the dataset consists of 5,000 complex questions. By extending the LC-QuAD framework to support Reified KGs and crowd-sourcing, we published the second version of the dataset as *LC-QuAD 2.0*, consisting of 30,000 questions with their paraphrases and has higher complexity and new variations in the questions. To overcome the problem of Entity Linking and Relation Linking in KGQA, we develop *EARL*, a module performing these two tasks as a single joint task for complex question answering. We develop approaches for this module, first by formalizing the task as an instance of the Generalized Traveling Salesman Problem (GTSP) and the second approach uses machine learning to exploit the connection density between nodes in the Knowledge Graph. Lastly, we create another large scale dataset to answer verbalization and provide results for multiple baseline systems on it. The Verbalization dataset is introduced to make the system's response more human-like. The *NQS* based KGQA system was next to the best system in terms of accuracy on the QALD-5 dataset. We empirically prove that NQS is robust to tackle paraphrases of the questions. *EARL* achieves the state of the art results in Entity Linking and Relation Linking for question answering on several KGQA datasets. The dataset curated in this thesis has helped the research community to move forward in the direction of improving the accuracy of complex question answering as a task as other researchers too developed several KGQA systems and modules around these published datasets. With the large-scale datasets, we have encouraged the use of large scale machine learning, deep learning and emergence of new techniques to advance the state-of-the-art in complex question answering over knowledge graphs. We further developed core components for the KGQA pipeline to overcome the challenges of Question Understanding, Entity-Relation Linking and Answer Verbalization and thus achieve our research objective. All the proposed approaches mentioned in this thesis and the published resources are available at `https://github.com/AskNowQA` and are released under the umbrella project *AskNow*.

Acknowledgements

First and foremost, I would like to express my sincere gratitude to my supervisor Prof. Dr. Jens Lehmann, for providing me with an opportunity to pursue my PhD study under his supervision. His patience, perseverance and motivation taught me to be calmly headed even during failures and progress towards the research goals. In our various discussions, he guided me to find research gaps and thus, we were able to better shape this research work. I am also thankful to Prof. Dr. Sören Auer for being the second examiner and for providing insights during our various research presentations.

Dr. Sourish Dasgupta, my Master's Thesis supervisor, introduced me to the field of Semantic Web and encouraged me to continue researching the field of Question Answering. Thank you, Dr Dasgupta, for inspiring me to do a PhD.

I want to extend my sincere thanks to Debayan Benerjee, Gaurav Maheshwari and Priyansh Trivedi for not only co-authoring research papers but also for providing support and constructive criticism to improve my research work. I gratefully acknowledge the efforts of Debayan for keeping the demos online and managing the infrastructure.

I have been fortunate enough to be a part of the Smart Data Analytics group, providing a platform to connect with great research colleagues. Special thanks to Gezim Sejdiu, Harsh Thakkar, Dr. Diego Esteves, Denis Lukovnikov, Dr. Giulio Napolitano, Debanjan Chaudhuri, Dr. Kuldeep Singh, Dr. Michael Galkin, Nilesh Chakraborty and Dr. Ricardo Usbeck. I further like to extend my thanks to Ankit Sharma, Debanjali Biswas, Konrad Höffner, Rony Hasan and Uttam Kumar who contributed to my thesis at various stages.

My family provided me with great strength with their patience and support during my studies. I would like to especially thank my elder brother Manish Dubey for keeping me free from all family responsibilities, which helped me to concentrate more on my studies. Last but not least, I thank my lovely and caring wife, Alpana, without whose support I could not have organized my research works so nicely. She provided me with continuous support during this phase of our life and encouraged me the most when times got rough. Thank you so much, dear.

Contents

CHAPTER 1

Introduction

Knowledge graphs can be used to curate and connect facts from a plethora of information sources available on the web. Potentially, a Knowledge Graph could comprise of complex facts which are challenging to gather manually and require experts of narrow domains. Retrieval of the complex information from a Knowledge Graph is an arduous task. Question Answering (QA) systems provide an interface to pose a question in natural language which in turn can retrieve information from knowledge acquired from complex data. Over the years, we have seen the emergence of Question Answering over different information mediums such as Wikipedia articles, Knowledge Graphs, news article, images and videos, extended to answering reading comprehension questions. Knowledge Graph Question Answering (KGQA) allows answering questions with complex facts. These composite facts require deep inferencing over the interlinking of data in a Knowledge Graph (KG). KGQA exempts users from two main access criteria to the Knowledge Graphs; first, the knowledge expertise of formal query language like SPARQL, and second, the specific vocabulary of the Knowledge Graph which they wish to inquire.

Looking at the history of Question Answering, the first use case of QA systems dates backs to 1960s. The QA system named Baseball [1] allowed users to frame a question in the English language about the US baseball league for that year. It answered user questions by performing syntactic analysis and dictionary lookup. Another domain restricted system LUNAR [2], showcased at a lunar science convention in 1971, effectively answered questions about the geological analysis of rocks from the Apollo moon missions. In the 2000s, MULDER [3] answered the general-purpose questions based on document ranking and used parse trees in building up the query. In the past decade, with adoption and widespread application of Knowledge Graphs, the research community has made significant advancement in the task of Knowledge Graph Question Answering. One of the first approaches for KGQA was a template-based solution [4] with pre-configured slots for the question or query. These rule-based approaches are still one of the conventional methods in KGQA. With improvements in Deep Learning techniques, the Question-Answering systems are also evolving from rule-based to more automated data-driven methods [5].

Recently, there has been a rise in Goal Oriented Dialogue Systems and Virtual Assistants such as Alexa [1], Bixby [2], Cortana [3], Google Assistant [4], Siri [5] and others. Current Smart Virtual Assistants are not limited only for doing repetitive tasks like making calendar entries, but they go far beyond and provide informative answers to the users' questions. Knowledge Graph Question Answering plays a pivotal role

[1] `https://developer.amazon.com/en-US/alexa`
[2] `http://bixby.samsung.com`
[3] `https://www.microsoft.com/en-us/windows/cortana`
[4] `https://assistant.google.com`
[5] `https://www.apple.com/siri/`

in the process of providing these intelligent features to a virtual assistant. Knowledge Graph stores both personal preferences and information from the web in a structural and interconnected way. Thus, Virtual Assistants can make more informed decisions and answer the question based on the Knowledge Graph.

Before proceeding to the research objective of this thesis, it is beneficial to understand the general workflow of Question Answering over Knowledge Graphs. In this context, one may look at three different aspects of KGQA which impact the final performance of the system (as shown in Figure 1.1). These are KGQA pipeline, KGQA datasets and the Knowledge Graphs.

The first facet is the KGQA component set and the pipeline [6], as they primarily constitute the system. Natural language used by humans can be complex, ambiguous, and not aligned with the KG vocabulary or the structure of the KG. Dependable KGQA systems require multiple steps and modules to capture human language nuances and align them with the Knowledge Graph. A generic QA pipeline may have modules for Natural Language Understanding (NLU), Named Entity Recognizer (NER), Named Entity Disambiguation (NED), Relation Linking (RL), Query Builder (QB) and Answer Representation (AR). For example, in the question "Where was the owner of the tesla born", the NLU module will understand the structure of the input and would provide insight on the semantics of the question. The NER module would identify the phrase "tesla" as the entity, and the NED component will connect "tesla" to "Tesla Inc." instead of linking it with "Nicola Tesla". Further, the RL module would link the two relations ("owner" and "birthplace") mentioned in the question. Now, utilizing all this information, the QB module would look into the KG and find the most suitable path in the KG for the user's question. Hence, the QB module makes a formal query for the question. The KGQA system retrieves the answer to the formal query, which is the requested answer of users' question. And finally, the answer is presented to the user.

The second aspect to look into is the dataset available for the KGQA task. Datasets help to improve the accuracy of individual modules, tasks and overall system. Current QA components are more robust and accurate due to modern techniques as compared to older rule-based approaches. One may categorize questions for KGQA as Simple questions and Complex Questions. Simple questions only have one entity and one property with a fixed formal structure. In contrast, Complex Questions have a significantly high amount of variations with multiple entity and property in a question and several formal structures. With the advent of large-scale datasets, KGQA over Simple questions has made significant progress by machine learning-based approaches [7]. Now, with the availability of large scale datasets, we expect similar progress for KGQA with Complex Questions. Researchers have applied template-based approaches to solve Complex Question Answering, with some success. The pre-recorded templates govern the performance of a template-based solution and creating more templates require manual efforts from domain experts. The research community is looking forward to having more generic and generative solutions to this problem and diverge from template based solution.

The third key facet is the Knowledge Graph itself. While a Knowledge Graph such as DBpedia relies on extracting facts from info-boxes in Wikipedia, Wikidata gets the data directly from its wiki editors, surpassing DBpedia in the number of entities and triple. The Knowledge Graph schema or ontology dictates structure and vocabulary of the final formal expression (SPARQL in most cases), to which the KGQA system maps the user question. With the introduction of Reification of KGs, now a triple could have metadata or other property related to the existing triples in the KG. Wikidata, YAGO and DBpedia have adopted these changes in their KG schemas. A KGQA system is bounded in its ability to answer based on knowledge acquired by the KG and quality of data in it [8, 9]. Hence, it is essential to consider these two factors while designing a KGQA system from an end-user perspective.

Research Objectives: The scope of this work is limited to Question Answering over Knowledge Graphs. Question Answering with Simple questions has already made significant progress with the availability of large-sized datasets and has achieved high accuracy as a task. However, the performance of the QA systems over Complex Question still has low accuracy.

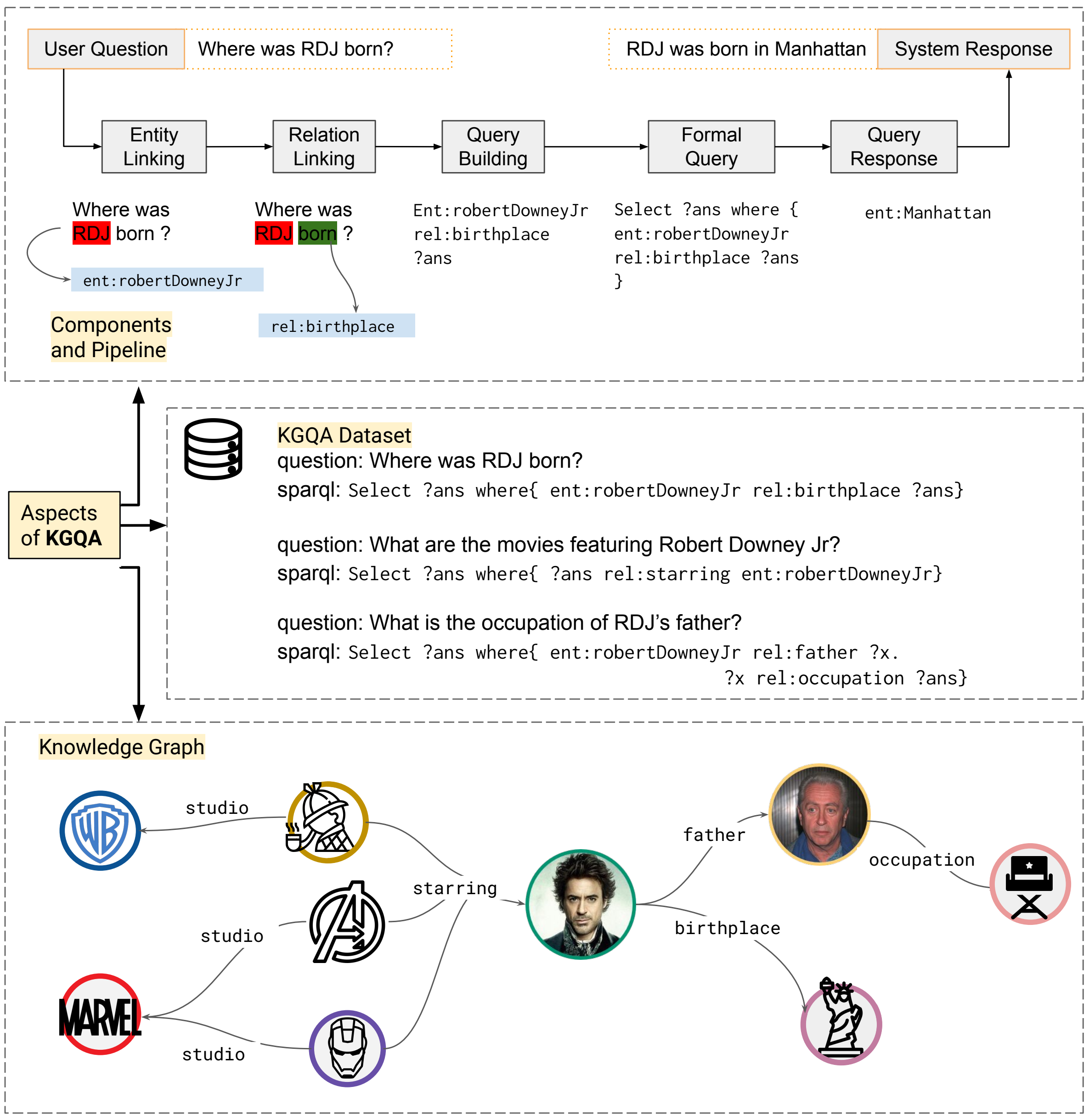

User Question
Where was RDJ born?
RDJ was born in Manhattan
System Response
Entity Linking
Relation Linking
Query Building
Formal Query
Query Response
Where was RDJ born ?
ent:robertDowneyJr
Where was RDJ born ?
rel:birthplace
Ent:robertDowneyJr rel:birthplace ?ans
Select ?ans where { ent:robertDowneyJr rel:birthplace ?ans }
ent:Manhattan
Components and Pipeline
Aspects of KGQA
KGQA Dataset
question: Where was RDJ born?
sparql: Select ?ans where{ ent:robertDowneyJr rel:birthplace ?ans}
question: What are the movies featuring Robert Downey Jr?
sparql: Select ?ans where{ ?ans rel:starring ent:robertDowneyJr}
question: What is the occupation of RDJ's father?
sparql: Select ?ans where{ ent:robertDowneyJr rel:father ?x. ?x rel:occupation ?ans}
Knowledge Graph
studio
studio
studio
starring
father
occupation
birthplace

Figure 1.1: Introducing the Three Facets of Question Answering over Knowledge Graph

Our objective in this work is to advance the state of the art in Question Answering over Complex Question such that the user gets a satisfactory response to the posed question. KGQA for Complex Question is more challenging than Simple questions in the sense that it would require us to handle this problem on multiple ends. The first step towards our objective is to design a natural language module which understands the user question. Then a large scale dataset for Complex-Questions needs to be generated. The next step is to develop natural language processing components, specifically for the KGQA. And finally, to have an approach to provide a human-like response to the user.

1.1 Motivation

Knowledge Graphs have the capability to store complex facts in the form of interlinked nodes which are not exploited to their full potential by Simple Questions. The motivation for this work is to improve the accuracy of KGQA over Complex Questions, so that the end-user of the system could explore complex information from the KG.

Here is an example of a simple question: "What's the profession of Robert Downey Jr. ?" or "What was the occupation of RDJ?". These questions could be answered based on a single triple in Knowledge Graph (such as Dbpedia, Wikidata and Freebase). Simple KGQA requires finding one entity and one relation in the input question. The formal query, in this case, remains static to a single pattern, so it does not pose a significant challenge. A Complex Question refers to more than one fact in the Knowledge Graph for retrieving an answer. For example, "What is the occupation of the father of RDJ?", here the system needs to identify one entity and two relations. In other Complex Questions, the requirement may be to identify more than one entity or relation. The formal query structure in the case of Complex Question is also not fixed, and instead, we find a significant variation in the query structure. Thus, Complex KGQA is relatively a difficult problem to solve.

For Simple Questions, there are large datasets such as SimpleQuestions[10] and 30M Factoid Question[11]. These datasets provide 100 Thousand and 30 Million questions, which are an ample amount of data to introduce machine learning and deep learning to this task. Other QA datasets such as WebQuestions[12] have fewer Complex Questions, as a majority of them (more than 90%) are simple questions. There are regular QALD [6] releases with datasets for various QA challenges. These datasets have an excellent mixture of Simple Questions and Complex Questions, yet the largest dataset published in QALD-9 [13] is less than 600 questions in size. Hence, there has been a lack of large scale datasets to enhance the performance of Complex KGQA.

Historically, the modules used in KGQA pipeline have not been designed for KGQA specific tasks and instead are generic to other NLP tasks. Consider the module Entity Linking (EL) which includes both Named Entity Recognition (NER) and Named Entity Disambiguation (NED). An Entity Linking module generally begins with span detection via NER, to generate candidates for each span, and then perform disambiguation (NED). Most EL modules rely on multiple entities in the text and then try to disambiguate mutually, say, by finding the shortest connection between the candidates in the Knowledge Graph. In questions, more often then not, we have only one entity in the question; thereby making it difficult for the EL system to disambiguate the entity due to lack of context. However, every question has at least one relation connected to the entity, which can potentially be utilized for better entity disambiguation. With a better Entity Linking system, we could advance state of the art in Complex Question Answering.

Natural Language Understanding (NLU) plays a vital role in KGQA for the sake of understanding the semantics of the questions. Natural language questions have high syntactic and semantic variations. Especially in Complex Questions, as here the formal query has more structural variety when compared to Simple Questions. The Knowledge Graph Schema further adds to the complexity of the task, as it extends the set of varieties of the formal query. We need a module which can extract the user intent and mentions among all the paraphrases of a particular question. These questions could have variations which are semantic equivalents; thus, the retrieved user input and user intent should be equivalent. Such an NLU component will make a QA system robust in terms of paraphrases of a question.

[6]QALD website: `http://qald.aksw.org`

1.2 Problem Statement and Challenges

In this section, we define the problem definition for this thesis work; then we look into the challenges posed by the problem definition. We provide examples for each challenge to give a better overview.

Research Problem Definition

How to improve the accuracy of translating the users' complex natural language questions to formal queries, to retrieve answers from the Knowledge Graph, and present the system responses effectively back to the user?

We recognize six fundamental challenges to be tackled while working towards our research problem, based on the motivation explained in the motivation section (Section 1.1 and Figure 1.1).

Challenge 1: Understanding the Intents of the User From the Posed Question

When retrieving the answer for a Complex Question, understanding the question is the primary task. Most of the QA systems do not try to understand the semantics of the user question and directly map a formal query against the natural language question [4]. This task becomes more challenging when dealing with multiple relations and entities in question.

Say, we have a question "What is the occupation of RDJ's father?". Here, the QA system needs to understand the semantics of the question. The NLU module needs to identify that there are one entity and two relations in the question. The NLU module faces the challenge of finding the dependency between the entity and relation. Here, the NLU module should connect occupation to father, and father to RDJ (mentioned entity).

Challenge 2: Lack of Large Size Complex KGQA Datasets

To improve upon the state of the art of KGQA on Complex Questions with a wide variety, we need a large dataset so that KGQA systems can be adequately trained and tested on them. In the past years, the research community released a few KGQA datasets. These datasets are either on simple questions [10] or do not have a high number of Complex Questions. QALD Challenge provides Complex Question datasets, but the size of these datasets is too small, as the questions are framed manually by KGQA researchers, and it is a time-taking activity. WebQuestion [14] dataset also provides Complex Questions, but out of the total 5000 questions, only 100 are Complex-Questions, and the rest are Simple-Questions. Creating a dataset having Complex Questions with a formal expression requires high manual efforts from domain experts. Thus we have a scarcity of significant scale Complex Questions datasets.

Challenge 3: Adaptation of Reification of Knowledge Graph into KGQA

Knowledge Graphs curates the data from the web and stores it in the triple format in a concise manner. In this process, they often do not store the details of the information; thus, a KG could consist of triples which are unclear or ambiguous. The Reification of KG adds metadata to triples mostly in the form of property-value pair and provides additional context for a triple, refer to example from Figure 1.2. In a non Reified KG, we could have a set of triple, which is misleading as the metadata information is not available. Whereas Reified KGs would have the extra context, which makes the semantic triple not

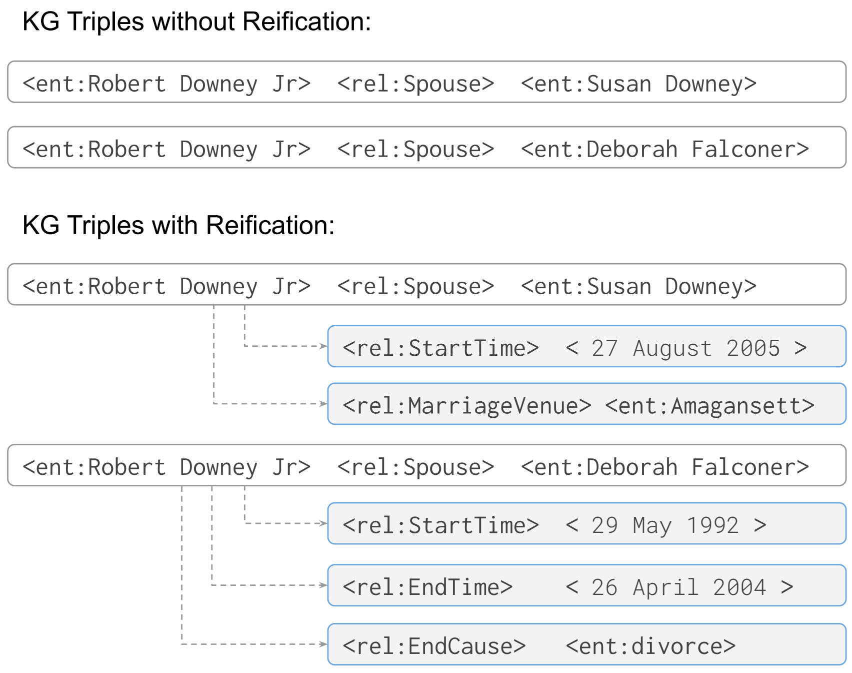

KG Triples without Reification:
<ent:Robert Downey Jr> <rel:Spouse> <ent:Susan Downey>
<ent:Robert Downey Jr> <rel:Spouse> <ent:Deborah Falconer>
KG Triples with Reification:
<ent:Robert Downey Jr> <rel:Spouse> <ent:Susan Downey>
<rel:StartTime> < 27 August 2005 >
<rel:MarriageVenue> <ent:Amagansett>
<ent:Robert Downey Jr> <rel:Spouse> <ent:Deborah Falconer>
<rel:StartTime> < 29 May 1992 >
<rel:EndTime> < 26 April 2004 >
<rel:EndCause> <ent:divorce>

Figure 1.2: Showcasing the effects of Reified KG Triples.

only evident in terms of understanding, but the KGs have more depth in their acquired knowledge. The adaptation of reified statements to Knowledge Graph is an unexplored area in KGQA research. Reified KG will allow the KGQA systems to produce an improved quality answer for the user questions. In the current state, there is no Question Answering system which uses this metadata. The KGQA community is posed with a challenge to adapt the Reified KG model in Question Answering research.

Challenge 4: Low Accuracy of Entity Linking Systems on KGQA

The Entity Linking modules used in KGQA system have lesser accuracy on QA dataset compared to large text datasets. Generally, to develop and train an Entity Linking system requires large text corpora with annotations. By design, these Entity Linking systems primarily perform entity disambiguation by taking the other surrounding entity into context. But, questions for KGQA usually have only one entity. Thus, the Entity Linking system fails to build a background for context-based disambiguation, and this causes a drop in the accuracy of the system.

For example, we have the two questions, "Who is the father of RDJ" and "What is the population of RDJ". The entity key phrase in both the questions is the same -"RDJ" [7]; in the first question we expect an Entity Linking tool to link ‘RDJ’ to ’Robert Downey Junior’ [8] the movie actor, whereas in the second question we require ‘RDJ’ to be disambiguated to ‘Rio de Janeiro’[9] a city in Brazil.0

[7]RDJ disambiguation: `https://en.wikipedia.org/wiki/RDJ`
[8]Robert Downey Junior: `https://www.wikidata.org/wiki/Q165219`
[9]Rio de Janeiro: `https://www.wikidata.org/wiki/Q8678`

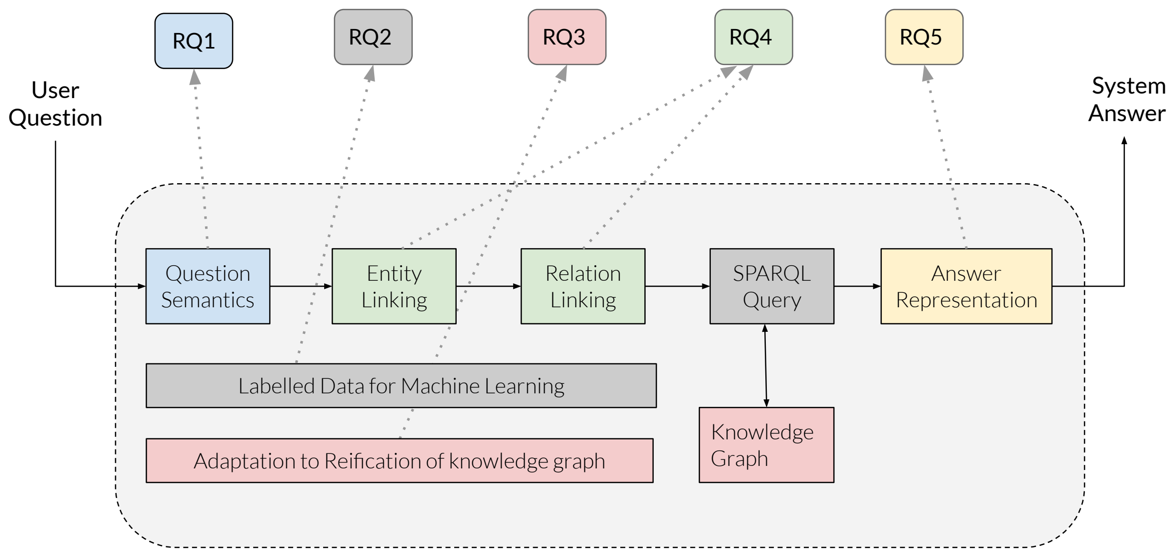


Figure 1.3: Demonstrating the connection of Research Questions to Question Answering Pipeline

Challenge 5: Low Accuracy of Relation Linking in KGQA

Relation Linking is one of the challenges not only in Question Answering but in general for NLP. Relation Linking task in KGQA is to identify the relation in the question and map it to a predicate in the KG. There are countless possibilities of phrasing and expressing a particular relationship. Let's look at a few examples, "What are the movies starring RDJ?", "Show movies where RDJ acted in?". These examples show the different semantics and syntactic variations in which a relation ("starring" in our case) could be expressed. Thus, Relation Linking is a challenging problem to solve. Moreover, a relation in a question could be implicit, where the relation phrase is hidden in the question. For example "movies in which we can see RDJ", here the relation "starring" is hidden in the question. This further increases the difficulty of the problem.

Challenge 6: Difficult to Capture the Context of the Question for Answer Verbalization

KGQA systems map the natural language question to a formal query and retrieve the answer. As a response to the user, the KGQA system gives the results retrieved by the formal query. The user feels unsatisfied if the system response is just the label of the retrieved answer, as the system response is not human-like and lacks context. For example, if the user's question is "Where was RDJ born?" and the system response is "Manhattan" , the user feels lack of context. But if the system response is "RDJ was born in Manhattan" the user feels satisfied as the answer is now more human-like. Some systems have employed template-based response to a user question, but this cannot be scaled to a large as Knowledge Graph having more than thousands of predicates and KGQA deals with a high number of Complex Questions with an enormous variety present in the natural language.

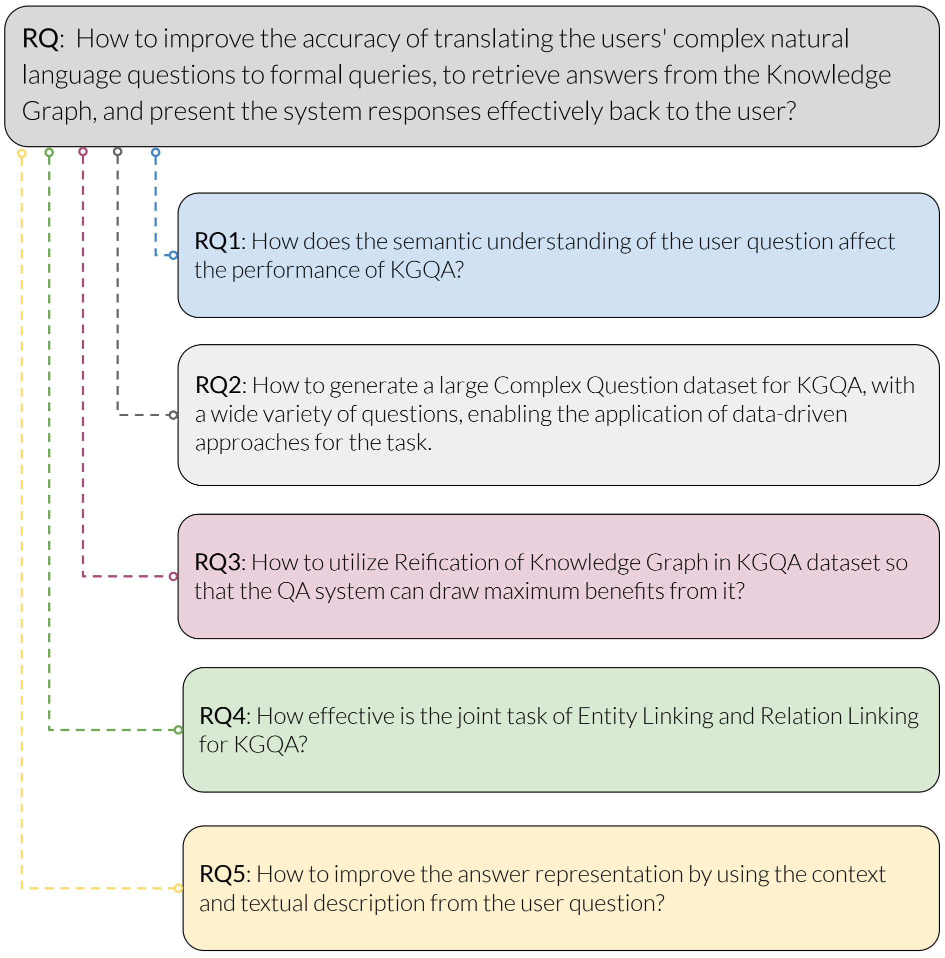

RQ: How to improve the accuracy of translating the users' complex natural language questions to formal queries, to retrieve answers from the Knowledge Graph, and present the system responses effectively back to the user?
RQ1: How does the semantic understanding of the user question affect the performance of KGQA?
RQ2: How to generate a large Complex Question dataset for KGQA, with a wide variety of questions, enabling the application of data-driven approaches for the task.
RQ3: How to utilize Reification of Knowledge Graph in KGQA dataset so that the QA system can draw maximum benefits from it?
RQ4: How effective is the joint task of Entity Linking and Relation Linking for KGQA?
RQ5: How to improve the answer representation by using the context and textual description from the user question?

Figure 1.4: Demonstrating the connection of Four Research Questions to Main Research Question

1.3 Research Question

Research Question 1 (RQ1)

How does the semantic understanding of the user question affect the performance of KGQA?

The enormous amount of data on the web attracts humans to fetch a variety of information every second. People not only try searching this web of data but also pose questions to fetch factual information. The first step to answer a question is to understand the question. This not only applies to machines but also to humans. A KGQA system needs a robust NLU module which interprets the question to help in making the final formal query expression.

Besides, humans can phrase the same question in different ways. In such a case, the question is structurally different but still holds the same semantics. This module needs to be tolerant of paraphrasing a question. Thus the different paraphrases of a question should have the same interpretation by the question understanding module. Besides, this module should be knowledge-graph agnostic.

Research Question 2 (RQ2)

How to generate a large Complex Question dataset for KGQA, with a wide variety of questions, enabling the application of data-driven approaches for the task.

In recent years, machine learning, specifically deep learning techniques, have achieved an advanced state of the art in many NLP sub-fields and beyond. These techniques often require a large dataset so that the machine may learn by observing various patterns in the data. With deep learning models on a large scale simple KGQA dataset, we have already witnessed considerable advancement in this sub-field of KGQA. Thus, a large size KGQA dataset with Complex Questions is a crucial requirement for the field to advance. Generation of such a dataset has only been possible at a small scale as such data curation requires supervision and manual effort by domain experts. To address this research problem, we need to generate large scale Complex Questions for Knowledge Graph in a way that the experts require minimal efforts. This strategy must address different varieties of Complex Questions.

Research Question 3 (RQ3)

How to utilize Reification of Knowledge Graph in KGQA dataset so that the QA system can draw maximum benefits from it?

The Knowledge Graph schema is one of the major dependencies in the KGQA pipeline. In recent years, Reification of Knowledge Graphs has brought advancement in the schema to support encapsulation of more detailed information. The traditional schema was able to hold a semantic triple without metadata related to the fact in the triple. Reified KG model offers to represent the metadata information for the triple and thus gives more details about a fact captured in the semantic triple. The KGQA community needs to adopt this shift in KG models to utilize the potential of the reified statements.

Research Question 4 (RQ4)

How effective is the joint task of Entity Linking and Relation Linking for KGQA?

We then explore the possibility and methodology to create a module which performs Entity Linking and Relation Linking as a common joint task. Traditional KGQA systems present the EL and the RL as individual tasks. These two tasks are either performed as a parallel task or as a sequential task. In a parallel setting, both the tasks do not share any information, hence lack in the local context. In sequential task settings, the error in one task cascades to the other task. Thus we examine to do these tasks as a joint task, where they share the context and improve the accuracy over the task.

Research Question 5 (RQ5)

How to improve the answer representation by using the context and textual description from the user question?

Answer representation is a crucial aspect in Question Answering from an AI perspective as we want more human-like response from the machine. Current Question Answering systems return answers directly from the results retrieved by the formal query. The goal of the QA system is to have a response which is more human-like. At present, response-answer most often lacks a context of the question. This area has been under-looked and accompanied by a limited amount of research work.

1.4 Thesis Overview

This chapter highlights the primary contributions to the dissertation. Here we refer to the scholarly articles covering those contributions published during the research period. We present an abstract outline and outcomes obtained through the conduct of the studies.

1.4.1 Contributions

Contributions for RQ1

An intermediate language that supports in understanding natural language questions and supports the generation of the corresponding formal query.

To address the Research Question RQ1 of understanding the user question, we propose an intermediate grammar named Normalized Query Structure (NQS) [15]. The NQS acts like an intermediate language in the process of translation of natural language question to a formal query. NQS works like a Natural Language Understanding (NLU) module for the questions. We develop a well-defined strategy to extract the semantics of the natural language question and express them into a formal query. Since the target Knowledge Graph is RDF/RDF-S in our scope, we express the formal query as SPARQL.

For a given question, "What is the occupation of RDJ's father?", NQS will be able to provide an understanding of the underlying semantics of the example question. NQS will indicate that the user intends to know about the 'occupation', and the user provides input as 'RDJ's father'. By breaking down the question-input, the phrase 'father' is indicated as a modifier to the question-input 'RDJ'. Thus automated construction of the formal query uses the semantic information marked by NQS.

Contributions for RQ2

A framework for generating complex natural language question with the corresponding formal query, in a semi-automated process.

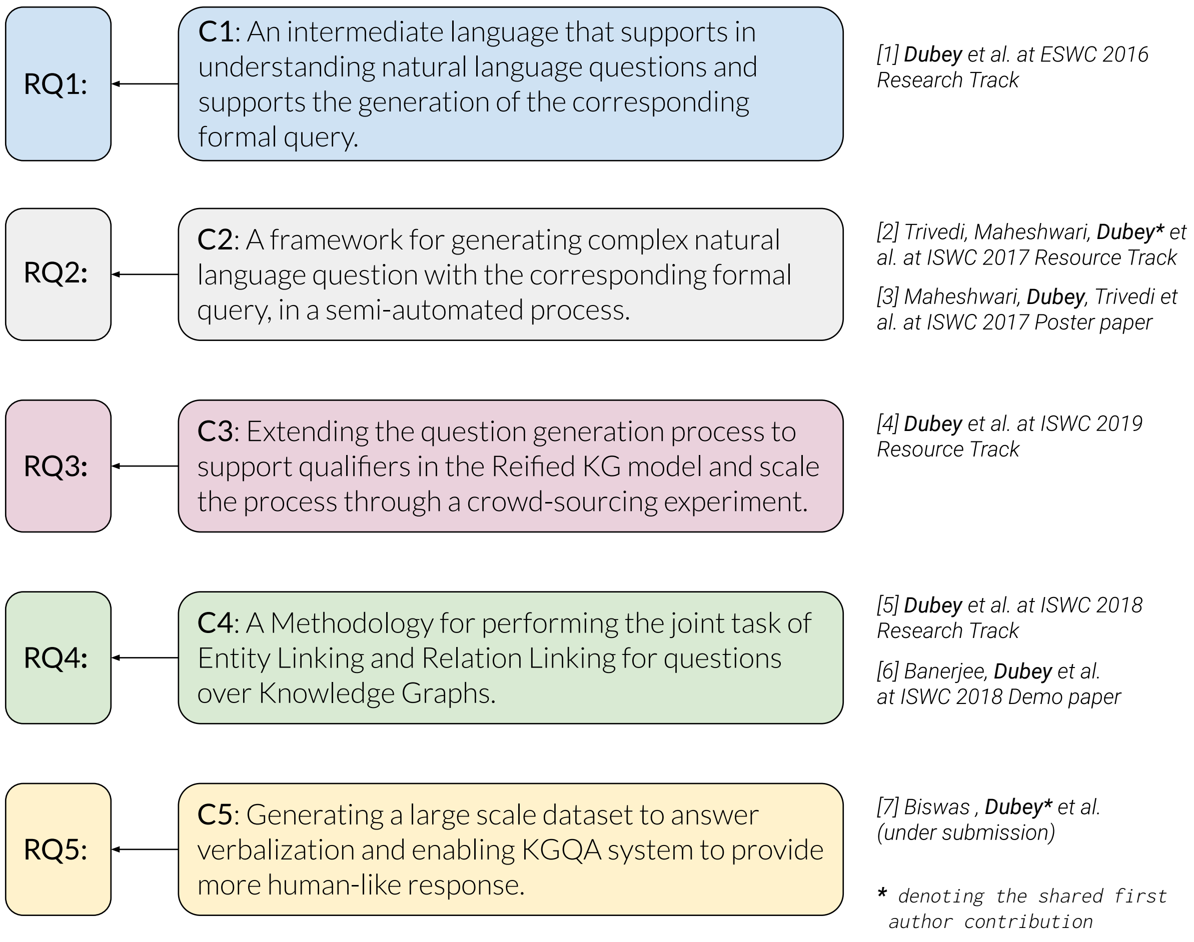

RQ1:
C1: An intermediate language that supports in understanding natural language questions and supports the generation of the corresponding formal query.
[1] Dubey et al. at ESWC 2016 Research Track
RQ2:
C2: A framework for generating complex natural language question with the corresponding formal query, in a semi-automated process.
[2] Trivedi, Maheshwari, Dubey* et al. at ISWC 2017 Resource Track
[3] Maheshwari, Dubey, Trivedi et al. at ISWC 2017 Poster paper
RQ3:
C3: Extending the question generation process to support qualifiers in the Reified KG model and scale the process through a crowd-sourcing experiment.
[4] Dubey et al. at ISWC 2019 Resource Track
RQ4:
C4: A Methodology for performing the joint task of Entity Linking and Relation Linking for questions over Knowledge Graphs.
[5] Dubey et al. at ISWC 2018 Research Track
[6] Banerjee, Dubey et al. at ISWC 2018 Demo paper
RQ5:
C5: Generating a large scale dataset to answer verbalization and enabling KGQA system to provide more human-like response.
[7] Biswas , Dubey* et al. (under submission)
* denoting the shared first author contribution

Figure 1.5: Contributions to the Research Question.

The advancement of the state of the art of KGQA, specifically Complex Questions, requires a large scale dataset with Complex Questions. Getting such a dataset created by non-experts is challenging, as it is hard for them to understand the interpretation of the formal query language. On the other hand, the process of dataset creation would require a lot of manual effort from the field experts. Thus, a large scale dataset for Complex Question in the past has been challenging to achieve. We were required to develop a framework where we reduce efforts of the field experts and generate Complex Question with their corresponding formal query.

Our proposal constitutes a Large-Scale Complex Question Answering Dataset (LC-QuAD) framework[16]. We generated a dataset of 5000 questions with their intended SPARQL queries for DBpedia. These questions exhibit significant syntactic and structural variations. The primary objective of LC-QuAD framework is to reduce the need for manual intervention from the field experts while producing Natural language questions and their SPARQL queries. In LC-QuAD, we first fetch a sub-graph around a seed entity and a white list of predicates. Given the sub-graph, we fit it into the sparql template to create a sparql query. We define a rule-based translation of a sparql template to natural language template named Normalized Natural Question Template (NNQT). NNQTs are then manually transformed into natural language question, provide a choice to vary the syntactic structure and semantics of the question. In the last stage, a reviewer examines the whole process to keep the quality of the dataset in check.

Contributions for RQ3

Extending the question generation process to support qualifiers in the Reified KG model and scale the process through a crowd-sourcing experiment.

With the wide acceptance of the reification of Knowledge Graphs, KGQA community also needs to adopt these changes in KG schema. Moreover, reification provides more valuable information in the Knowledge Graph, and that enables the QA system to cover more details and connected information for an entity. The use of qualifiers enables the Knowledge Graph to expand, annotate or get richer context for a statement and express information beyond what a simple classical triple could store.

For example, qualifiers can capture the temporal and spatial information of a statement. Thus, the use of qualifiers increases the overall knowledge to be retrieved from a KGQA system. In our work LC-QuAD 2.0 [17], we create another KGQA dataset which adapts the reified KG model. LC-QuAD 2.0 is the largest dataset of 30,000 Complex Questions with corresponding SPARQL queries for Wikidata and DBpedia 2018. In the sparql queries, we capture qualifier information and metadata related to a triple statement and create more informative Complex Questions in the dataset. We introduce several new categories of the question not present in any KGQA dataset. By using qualifier information, we frame questions which are related to meta-information for a semantic statement. For example 'Where did RDJ got married to Susan Downey?', here the statement is <RobertDowneyJr - spouse - SusanDowney>, and the question is on the metadata information 'location' attached to the statement. LC-QuAD 2.0 has a large number of questions with temporal aspects to them. In this dataset, the temporal information was not only in the question intent but also in the question input. The previous Knowledge Graph schema would not capture such information about the metadata. Moreover, here the dataset has questions with dual user intentions, for example, 'When and where RDJ was born?'. Here, the user inquires about two intents (desires) in the same question.

Contributions for RQ4

A Methodology for performing the joint task of Entity Linking and Relation Linking for questions over Knowledge Graphs.

Entity Linking and Relation Linking plays a pivotal role in KGQA. They perform the task of linking the question phrases to the Knowledge Graph, which is helpful in the formal query generation for a corresponding natural language question. Hence, it is crucial to perform the linking process with high accuracy, and this is a major bottleneck for the widespread adoption of current KGQA systems. In most Entity Linking systems [18, 19], disambiguation is performed by looking at other entities present in the input text. However, in the case of natural language questions (short text fragments), the number of other entities for disambiguation is not much. Therefore, it is potentially beneficial to consider entity and relation candidates for the input questions in combination. This is required to maximize the available evidence for the candidate selection process. To achieve this, we propose EARL (Entity and Relation Linker) [20], a system for jointly linking entities and relations in a question to a Knowledge Graph. EARL treats Entity Linking and Relation Linking as a single task and thus aims to reduce the error caused by the dependent steps. EARL achieved the state of the art results for Entity Linking and Relation Linking for KGQA at the time of publication.

Contributions for RQ5

Generating a large scale dataset to answer verbalization and enabling KGQA system to provide more human-like response.

Our goal here is to improve the answer verbalization such that the user experience for the QA system is enhanced. We develop a module which takes two inputs, the user's natural language question and QA system's formal query results and the module response back with a verbalized answer framing a natural language sentence. For such a task, most systems have employed template-based methods as there was no large scale dataset available to train a machine learning model. Thus, we create a large scale dataset with more than 100k questions with their verbalized answers. We also showcase that the answers are verbalized differently, in comparison to the RDF-triple verbalization. The verbalized answers have more context of the question by reusing the same vocabulary as in the input question, and thus, the response is more human-like. We further deployed the state-of-the-art natural language generation approaches to have baseline models working on our dataset.

1.4.2 Publications

The following publications constitute a scientific basis of this thesis and serve as a reference point for numerous figures, tables and ideas presented in the later chapters:

- *Conference Papers (peer reviewed)*
  1. **Mohnish Dubey**, Sourish Dasgupta, Ankit Sharma, Konrad Höffner, Jens Lehmann. "Asknow: A framework for natural language query formalization in sparql." In Extended Semantic Web Conference, pp. 300-316. Springer, Cham, 2016.
  2. Priyansh Trivedi, Gaurav Maheshwari, **Mohnish Dubey**, and Jens Lehmann. "LC-QuAD: A corpus for Complex Question Answering over Knowledge Graphs." In International Semantic Web Conference, pp. 210-218. Springer, Cham, 2017. The first three authors contributed equally.
  3. **Mohnish Dubey**, Debayan Banerjee, Debanjan Chaudhuri, and Jens Lehmann. "Earl: Joint entity and Relation Linking for Question Answering over Knowledge Graphs." In International Semantic Web Conference, pp. 108-126. Springer, Cham, 2018.
  4. **Mohnish Dubey**, Debayan Banerjee, Abdelrahman Abdelkawi, and Jens Lehmann. "LC-QuAD 2.0 A large dataset for Complex Question Answering over wikidata and DBpedia." In International Semantic Web Conference. Springer. 2019.
- *Demo Papers (peer-reviewed)*
  5. Gaurav Maheshwari, **Mohnish Dubey**, Priyansh Trivedi, and Jens Lehmann. "How to Revert Question Answering on Knowledge Graphs." In International Semantic Web Conference (Posters, Demos and Industry Tracks). 2017.
  6. Debayan Banerjee, **Mohnish Dubey**, Debanjan Chaudhuri, and Jens Lehmann. "Joint Entity and Relation Linking Using EARL." In International Semantic Web Conference (P&D/Industry/BlueSky). 2018.

- *Published Dataset as Resources*
  1. Priyansh Trivedi, Gaurav Maheshwari, **Mohnish Dubey**.
     "LC-QuAD 1.0 Dataset"
     Figshare: `https://figshare.com/projects/LC-QuAD/21812`.
  2. **Mohnish Dubey**,
     "Fully Annotated LC-QuAD 1.0 Dataset"
     Figshare: `https://figshare.com/projects/EARL/28218`
  3. **Mohnish Dubey**.
     "LC-QuAD 1.0 German version Dataset"
     Figshare: `https://figshare.com/projects/LC-QuAD_1_0_German_version/83657`
  4. **Mohnish Dubey**.
     "LC-QuAD 2.0 Dataset"
     Figshare: `https://figshare.com/projects/LCQuAD_2_0/62270`
  5. **Mohnish Dubey**, Debanjali Biswas
     VANiLLa dataset
     Figshare: `https://figshare.com/articles/Vanilla_dataset/12360743`

The whole list of publications completed during the PhD term is available in Appendix A.

1.5 Thesis Structure

The structure of the thesis consists of nine chapters. Chapter 1 introduces the primary research problem of the thesis and the motivation for the research undertaken. We look into the challenges to be faced for our research goal and drive research questions to overcome the obstacles. We systematically describe the research questions, contributions towards these research questions, and a list of scholarly publications based on the contributions. Chapter 2 provides insights into the preliminary concepts in the area of Semantic Web, Knowledge Graphs and Question Answering, required for understanding the research problem and selected approaches to address these research questions and contributions. Chapter 3 discusses state-of-the-art community efforts in various domains, e.g., Knowledge Graphs, Question Answering over Knowledge Graph, Entity Linking, Relation Linking. We also look into the different KGQA datasets and previous work done in Natural Language Generation. Chapter 4, for Question Answering systems, we define an intermediate language used for converting a natural language question to a formal query. We deeply investigate working of the question understanding module, and how it treats the information given in the user question. Chapter 5, defines the semi-automated approach to generate Complex Questions over Knowledge Graphs. We use this methodology to create an extensive dataset of 5000 Complex-Question with corresponding sparql queries. We showcase the impact of this dataset in KGQA community by enlisting KGQA modules and full systems, advancing state of the art using the dataset. Chapter 6, extends the work of chapter 5 and adopts the framework to Reified Knowledge Graph. Here we upscale the size of the dataset generated to 30,000 Complex Question paraphrases.Chapter 7, describes the KGQA module we developed to perform Entity Linking and Relation Linking as a joint task over questions. Here we discuss the two approaches for this joint task, one from the algorithm point of view and other from a machine learning perspective. Chapter 8 delves into the release of an answer verbalization dataset. Here we provide multiple baselines on our dataset. Finally, Chapter 9 concludes the thesis with the directions for future work.

CHAPTER 2

Background

For working on the research problem of having Question Answering system over Knowledge Graphs, it requires a comprehensive understanding of the fundamental of necessary fields such as Semantic Web and Knowledge Graph. We aim to understand the essential components used in the question answering system and how they impact the whole Question Answering pipeline. In this Chapter, we will first look into the Semantic Web and the evolution of the Knowledge Graph. We then look into the fundamentals as to why we need a question answering system. Then we dive deep in basics of Question Answering over Knowledge Graphs, by studying all the components required for a KGQA system.

2.1 Semantic Web and Knowledge Graph

The Web is a system where documents and web resources containing information are attached to a Uniform Resource Locators (URLs) that may be interlinked and are available over the Internet [21]. The Semantic Web is an extension to the Web to enable the machine to understand and process the information available on the Web. The goal here is to make the data express its meaning to machines so that the machine-agents could perform mundane tasks.

In the year 1996, we witnessed the invention of the Web. The initial use-case of the Web was to have static websites and documents published online. Fewer people were publishing on the Internet, and thus it was mostly a read-only web, later termed as web 1.0.

Later in the 2000s, the Web was introduced with new features like tags and annotations. Gradually the Web progressed to become a read-write service, and several platforms provided people to write information about a wide variety of topics, thus achieve a huge reader base. As a result, the websites became more interactive in nature. In a social media forum, Web 2.0 allows user-generated content to connect and communicate with each other in a virtual environment.

With the emergence of Semantic Web, we gradually migrated to Web 3.0. The Semantic Web technologies use declarative ontological language like OWL to generate domain-specific ontology. Such ontologies can support machines for knowledge reasoning and making new inferences and not just rely on string matching. The Semantic Web extends the Internet just from hyperlinks of URIs representing information in web-pages to have machine-understandable metadata about the web pages and URIs, giving more details about the interrelation of the two linked items.

The Semantic Web formally represents the metadata through technologies such as Resource Description Framework (RDF) and Web Ontology Language (OWL) World Wide Web Consortium (W3C) specifications for the model of metadata are given in the Resource Description Framework (RDF). It has been utilized as a general way to describe or model information in a variety of syntax notations

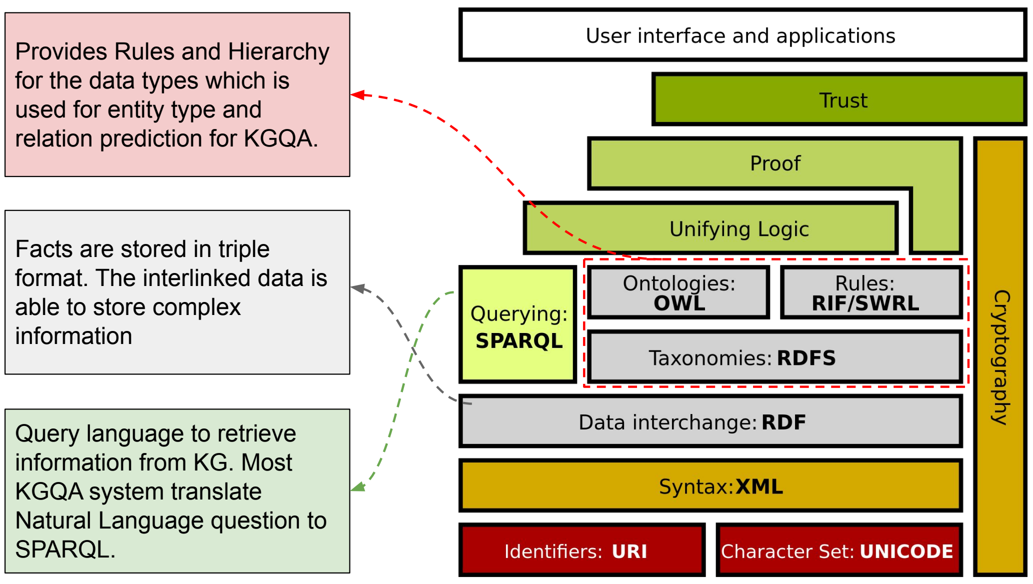

Provides Rules and Hierarchy for the data types which is used for entity type and relation prediction for KGQA.
Facts are stored in triple format. The interlinked data is able to store complex information
Query language to retrieve information from KG. Most KGQA system translate Natural Language question to SPARQL.
User interface and applications
Trust
Proof
Unifying Logic
Querying: SPARQL
Ontologies: OWL
Rules: RIF/SWRL
Taxonomies: RDFS
Data interchange: RDF
Syntax: XML
Identifiers: URI
Character Set: UNICODE
Cryptography

Figure 2.1: Mapping KGQA and Semantic Web Stack

and serialize data, which are carried out in web resources. It is also used in software for information management.

2.1.1 The Semantic Web Stack from a KGQA Perspective

Figure 2.2 is showing the Semantic Web stack, which includes the hierarchy of languages, data formats and concepts are used such that each layer utilizes the output of the layer stacked below it. The stack showcase about technology and community standards have been organized to enable the Semantic Web. It also demonstrates that Semantic Web is an extension and not a substitution to the classical Web.

Currently, the technology from the bottom of the Stack to OWL is standardized and approved for developing Semantic Web applications. Whether the top of the Stack should be handled is still uncertain. To achieve the full vision of the Semantic Web, every layer of the Stack has to be implemented. The Semantic Web stack could be categorized in three different parts, that is i)Hypertext Web technologies ii)Standardized Semantic Web technologies iii) Unrealized Semantic Web technologies.

At the bottom layers, we see the use of long term techniques from hypertext web and are directly used in the semantic Web. The bottom-most layer provides unique IDs to each resource on the Web by assigning Internationalized Resource Identifier IRI, a generalization of URI. The XML provides semantic meaning and interconnection to the semi-structured data. Unicode supports connecting documents across different languages spoken around the Web by humans. The third (upper) part of the Stack consists of technologies that have not yet been standardized. This part is built by Cryptography layer, User interface and Trust layer.

The second part (middle layers) consists of Semantic web technologies and standards, which are authenticated by W3C. This part of the Stack is responsible for providing the base to develop the semantic web application. As further discussed in this section question answering over knowledge graph is also highly connected to this part of the Stack. The second part of the Stack consists of a Resource Description Framework (RDF) which allows information about resources(from the bottom layers) to be represented as a graph. We then have RDF Schema (RDFS) which gives an underlying schema to RDF data. The RDFS is further advanced by OWL(Web Ontology Language) and RIF (rule interchange format) by using the basics of Description Logics (second-order logic) enabling the semantic web stack with reasoning capabilities. We have SPARQL which works as the query language.

From a question answering perspective, the middle part of the semantic web stack is essential. The QA systems connect to these layers for various functionalities which can be stated as follows: RDF layer stores those facts in triple formats. These triples have the information which the QA system retrieves for the user. Interlinking of data at this stage results in storing complex information which is not perceived in its raw format. RDFS and OWL layer provides explicit semantics to the information stored in RDF, and hence the data can now be inferred and reasoned, transforming the information into knowledge. Most QA systems use SPARQL as their formal query language, treating KGQA as a task of translating natural language question to sparql query. The ontology provided by the RDFS and OWL plays a vital role as it drives the semantics of the SPARQL. The semantics provided by ontology is also crucial while performing disambiguation for Entity Linking and Relation Linking, as the ontology provides Rules and Hierarchy for the data types.

2.1.2 Knowledge Graph

An essential task in AI is to transfer the worldly human knowledge to the intelligent systems so that they could learn to solve problems like humans. This requirement of knowledge representation makes knowledge graphs as a critical research area in AI not only for academia but also for industry. Knowledge graph stores the information about the real-world entities, events and concepts. For instance, an entity in the knowledge graph has a semantic description and further would have multiple properties connecting it to other entity or concepts or events.

The underlying structure of the Knowledge Graph(KG) is Graph, so let us first formally define a graph G, to get a better understanding of the knowledge graph. An intuitive definition of a graph is a pair consisting of a set of vertices and a set of edges.

Formalization: Graph

A graph is an ordered pair $G = (V, E)$ such that: V is a set, called the vertex set;
E is a set of 2-element subsets of V, called the edge set. That is: $E \subseteq u, v : u, v \in V$.

E can also be described as an anti-reflexive, symmetric relation on V.

For a KG, the entities are always on the nodes, and relations are on edges. Further, there could be literals on the leaf nodes which store string and numerical information. One of the most common practices for a KG construction is to align the facts about an entity to already existing ontology. Further from the literature, we can take the following two definitions of the KG and then formally define KG.

"[22] *A knowledge graph acquires and integrates information into an ontology and applies a reasoner to derive new knowledge.*"
"[23] *A KG is a multi-relational graph composed of entities (nodes) and relations (different types of edges). Each edge is represented as a triple of the form (head entity, relation, tail entity), also called a fact, indicating that a specific relation connects two entities.*"

RDF Triple

Definition 2.1.1 *Let $\mathcal{U}, \mathcal{B}, \mathcal{L}$ be disjoint infinite sets of URIs, blank nodes, and literals, respectively. A tuple $(s, p, o) \in (\mathcal{U} \cup \mathcal{B}) \times (\mathcal{U}) \times (\mathcal{U} \cup \mathcal{B} \cup \mathcal{L})$ is denominated as an RDF triple, where s is called the subject, p the predicate, and o the object. (s,p,o) is a generalized RDF triple when $(s, p, o) \in (\mathcal{U} \cup \mathcal{B}) \times (\mathcal{U}) \times (\mathcal{U} \cup \mathcal{B} \cup \mathcal{L})$. An element $\mathcal{U} \cup \mathcal{B} \cup \mathcal{L}$ is called an RDF term.*

RDF Graph [24]

Definition 2.1.2 *Let $\mathcal{U}, \mathcal{B}, \mathcal{L}$ be disjoint infinite sets of URIs, blank nodes, and literals, respectively. An RDF graph G is a directed labeled multigraph $G = (\mathcal{N}, E, \Sigma, \mathcal{L})$ where:*

- $\mathcal{N} \subset (\mathcal{U} \cup \mathcal{B} \cup \mathcal{L})$ *a finite set of RDF terms that correspond to nodes*
- $E \subseteq \mathcal{N} \times \mathcal{N}$ *a finite set of edges that connect RDF terms*
- $\Sigma \subset \mathcal{U}$ *a set of labels uniquely identified with URIs*
- $\mathcal{L} : E \to 2^{\Sigma}$ *that maps edges to sets of labels*

We further formally define the Knowledge Graph KG.

Formalization:Knowledge Graph

Within the scope of this thesis, we define a *knowledge graph* as a labeled directed multi-graph. A labeled directed multi-graph is a tuple $KG = (V, E, L)$ where V is a set called vertices, L is a set of edge labels and $E \subseteq V \times L \times V$ is a set of ordered triples.

It should be noted that our definition of knowledge graphs captures basic aspects of RDF datasets as well as property graphs [25]. The knowledge graph vertices represent entities, and the edges represent relationships between those entities.

The research directions and application of the knowledge graph [26] could be generalized into four categories, namely i) Knowledge Aware Application ii) Knowledge Acquisition iii) Knowledge Representation Learning and iv) Temporal Knowledge Graph. In Figure 2.2, we show the four categories with other research field associated to them. We also highlight the research area covered in this thesis.

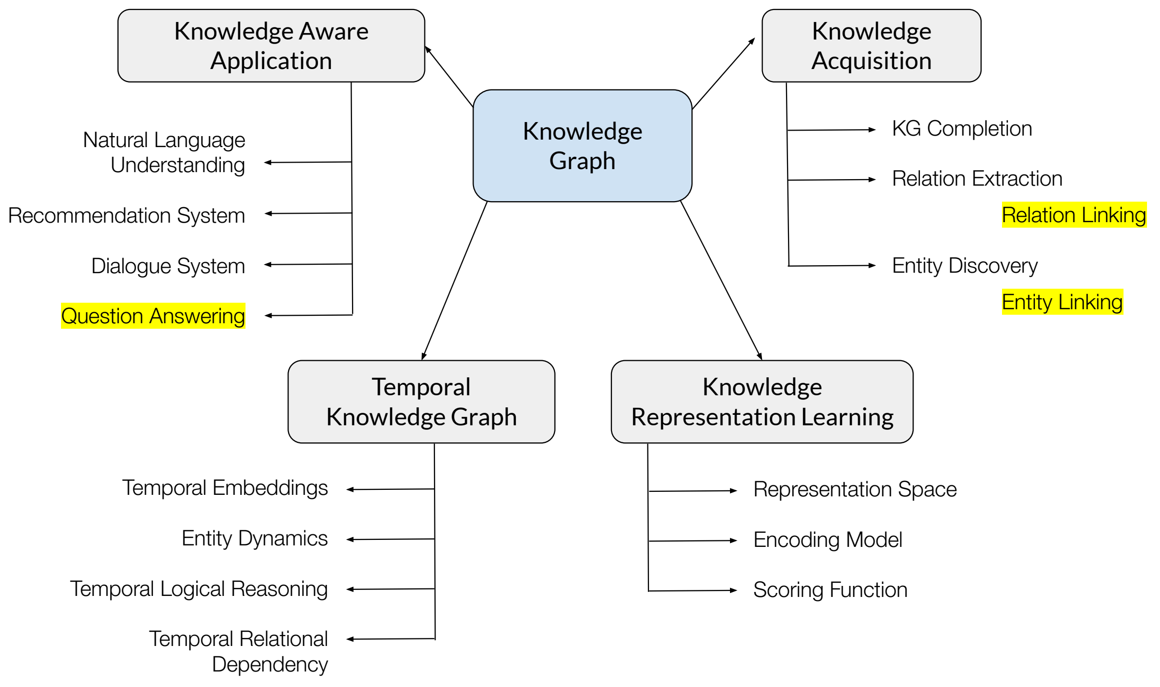


Figure 2.2: Categorization of Research Areas and Application of Knowledge Graphs (adopted from [26])

2.2 Why Question Answering and not Search

The arrival of search engines for the Internet, led users to choose keywords needed to find answers or gather information on a topic. A web search engine (internet search engine) offers a platform which programmatically searches the web for information described in a textual web search query or search keyword. Search engine outputs pages (SERPs), displays results in a line format containing a variety of web pages, images, videos, info-graphics, posts, research papers and other file types. If one is good at guessing the search keyword, the results are correct with the best answer ranked at the top. This setting works when the user aims to collect general information for the topics/entity mentioned in the search keyword. But if the user aims to find an answer to a specific question, the user must pass several URL, or results pages, to find the right answer. The user often has to search through the content to find an answer even when the search engine's results have a correct Web site or text. Searching is tedious, time-consuming, and often unreliable, whether you use a search engine or a website's search field.

According to the study 'Search Engine Use 2012' [27], in 2012, 56% of the Search users were 'very confident' in their search abilities as compared to the 48% users in 2004. The survey revealed that the Search users with at least college education or those with higher income brackets, were more confident in using the search engines. Moreover, 29% of the Search users were 'always' successful in getting the information they were searching on the search engine. Though this study is eight years old and the search engines have improved a lot since then, the fundamentals of the search systems remain the same.

Asking questions more natural to human language is then compared to a search term. In standard search engine techniques, the searched key phrase go through the process like TF-IDF [28], Stop word removal, and stemming from making the final search query phrase. This modified search query phrase is then used to gather data, alter user question entirely and lose the original intent (or desire). This hampers the final search results, and then the user is shown a whole list of result on SERP. The user is then

Question Answering	**Search Engine**
Questions are expressive and user intentions are clear.	Keyword search leaves space for ambiguity.
Questions come naturally to humans from the language they speak and write.	keyword search is an adaptation to the search system interface.
Used to fetch a fact for a particular entity.	Used to collect more information about the given input.
User-driven, a user asks for precise information	System-driven, a user is provided with information which system predicts would be useful for the user.

Table 2.1: Comparison of Question Answering and Search Engine

required to go through the manual process of finding the answer to the question in these search results. On the other hand, a QA system utilizes all the possible information mentioned in the user question. The question usually goes through the process where the question intention and answer type are acquired from the question itself. The final output to the user is a direct answer to the question asked by the user. For the end-user question, answering provides a more human-like experience.

2.3 Question Answering over Knowledge Graph

Question Answering over Knowledge Graph (KGQA) aims to retrieve the information intended in the question from the information stored in the knowledge graph. Traditional QA systems are a series of non-monolithic component working together. Over the years, researchers are working towards a monolithic QA system, where they have achieved considerable success by targeting to have end-to-end models for KGQA with simple questions. Here we showcase different components of the QA system

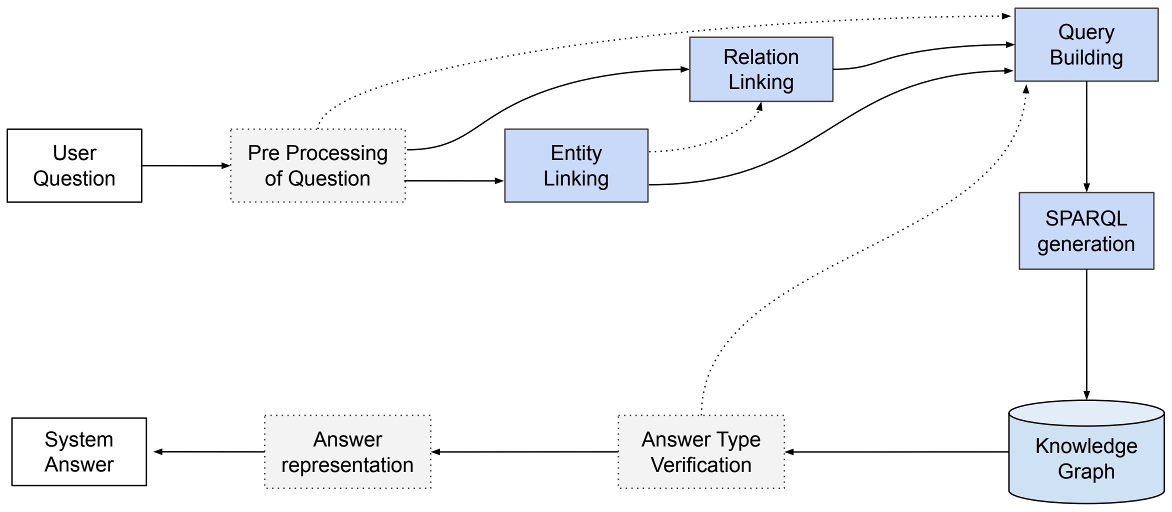


Figure 2.3: Components involved in KGQA

pipeline, which we have derived from generic QA architecture.

The pre-processing of input question string is standard practice by many QA system. It involves some fundamental natural language processing (NLP) operations, such as Shallow parsing, POS tagging. Entity Linking and Relation Linking connects the question to the knowledge graph. The query builder generates the query to retrieve the information intended in the original question. In systems, there is an answer verification procedure where they match the semantic answer type with the type predicted based on the question. At last, the answer is presented back to the user in appropriate representation; for example, in the dialogue system setting the answer could be verbalized into a sentence.

We further formalize the process of KGQA.

Formalization: Question Answering over KG

Given a natural language question Q_{NL}, assuming Knowledge Graph KG with a collection of Entity set E and Relation R. The KGQA system aims to retrieve Answer A of the question from the KG, by translating Q_{NL} to a Formal Query Q_{FL}, such that they are semantically equivalent.
Given a language question Q_{NL} KGQA aims to model a transformation function f_{kgqa} such that :

$$f_{kgqa} : Q_{NL} \mapsto Q_{FL}$$

$$(Q_{NL})^{I_{NL}} \equiv (Q_{NL})^{I_{FL}}$$

I_{NL} is the linguistic interpretation of the natural language question
$I_{FL}$is the semantic interpretation of the formal query.

We now further look into various KGQA components to get better background knowledge of our problem statement.

2.3.1 Entity Linking

Named Entity Recognition is the function of assigning a unique identity-class (such as a person, location, things or events) to the entities mentioned in the text. Entity Linking is the process to spot a mentioned entity in the text (question in our use-case) and connect it to the entity representing the span-entity in the knowledge graph.

With Entity Linking, we are able to use a large amount of information on the real-world individuals and their relationships in the publicly available knowledge graphs, such as Wikipedia, DBpedia or Wikidata. Entity Linking connects text (unstructured) data to a structured database, providing a base to use the knowledge stored in structured data. This gives more semantic information to understand the texts better. For example, in the sentence "Manhattan is the birthplace for Robert Downey Jr", by Entity Linking we can link "Robert Downey Jr" to `https://www.wikidata.org/wiki/Q165219` and "Manhattan" to `https://www.wikidata.org/wiki/Q11299`. These linkings shows that the sentence is regarding movie actor. We can find other facts in the knowledge graph regarding the actor(Example: Robert Downey Jr is married to Susan Downey) or things connected to the city (Example: Manhattan is next to river Hudson). This extension of semantic information for the entity could provide help in a wide range of applications including semantic search, question answering, semantic annotation, relationship extraction, text enrichment, entity summarization, etc.

The task of the Entity Linking(EL) module is to mark the entity in the sentences and then link them to the knowledge graph, respectively. In Question Answers, EL needs to identify the span of the entity and connect to the knowledge graph.

Formalization: Entity Linking

Given a natural language question (Q_{NL}) with String set $S = \{s_1, s_2, s_3...s_n\}$ where s is a word token, and n is number of words in the question. Assuming Knowledge Graph KG with collection of Entity $E = \{e_1, e_2, e_3, ..., e_m\}$, where m is a large number representing a total number of entity (e) in the KG, the Entity Linking system aims to identify all the entity mentions $M = \{M_1, M_2, ...M_m\}$ in S, and connect each M_i to respective e_i from E.

Entity Linking is generally a three-step procedure involving Entity Span(Mention) Detection, Candidate Generation and Candidate Disambiguation. The first task is to spot the span of the entity in the question string. In Span/Mention Detection phase, the entity linker needs to find the sequence of string = $s_i...s_j$ which could represent a named entity (e). Once the *span* of the entity is detected, the EL system applies string similarity to generate a set of Entity Candidate E_{cand} for the span, such that $E \subset E_{cand}$. The final task is to re-rank the E_{cand} list such that the correct candidate is the top of the list.

The following are challenges [29, 30] that need to be overcome by the Entity Linking module.

- Disambiguation: This is one of the critical challenges in Entity Linking. Let us take the following question as an example: "Jaguar is the product of which company?". Here the word Jaguar is correct token span for Entity Linking. The entity candidates with the same labels are Jaguar - the animal, the car, the title of the song. In this case, the disambiguation needs to take more context from other parts of the example sentence (i.e. product, company) to reach the correct entity(Jaguar the car) in the KG. There are hard ambiguities too in Entity Linking, which could be very hard even for humans to disambiguate. For example "Where was John born ?" here "John" is the entity

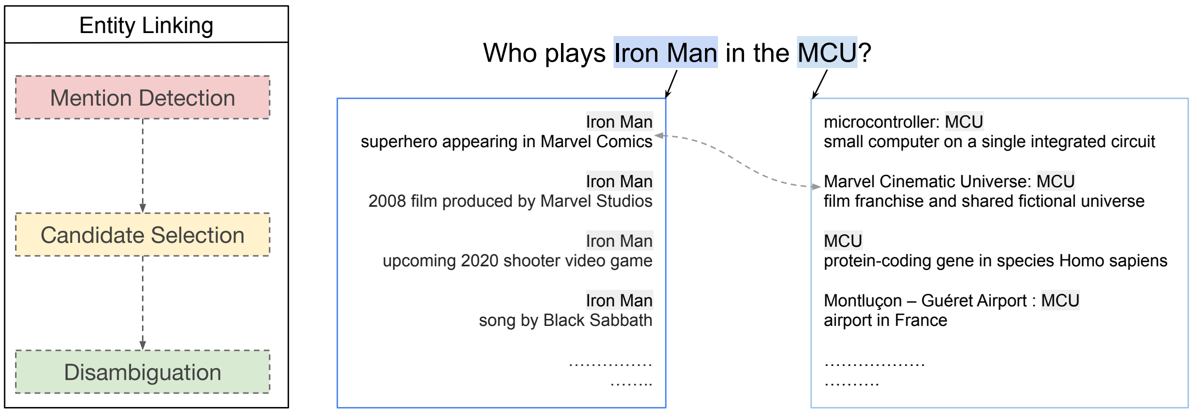

Entity Linking
Mention Detection
Candidate Selection
Disambiguation
Who plays Iron Man in the MCU?
Iron Man
superhero appearing in Marvel Comics
Iron Man
2008 film produced by Marvel Studios
Iron Man
upcoming 2020 shooter video game
Iron Man
song by Black Sabbath
microcontroller: MCU
small computer on a single integrated circuit
Marvel Cinematic Universe: MCU
film franchise and shared fictional universe
MCU
protein-coding gene in species Homo sapiens
Montluçon – Guéret Airport : MCU
airport in France

Figure 2.4: Steps in Entity Linking

span but its very hard to decide which John to choose for the entity in the KG as there are multiple people (entity) with John as their name(label).

- Incorrect case: For correct English grammar when we mention a real-world entity (mostly proper noun), we start the word with an upper case. The large corpora of text with the manually annotated entity are used for training entity recognizer. These NER models (Stanford, spacy) often learn patterns from natural language grammar. Thus when the test input is of wrong grammar such as incorrect case, they fail to recognize the entity.

- Multi-word entity: there are entities which have long names such as the title(label) of a book, movie, song or building names. These long names are usually longer than three words and their appearance in web text are skewed compared to say the name of a person. These multi-word entities are challenging to recognize for the entity span. For example "Who is the author of My Experiments with Truth?", where 'The story of My Experiments with Truth' is the correct entity span, but most NERs fail to recognize it.

- New aliases: Most Entity Linkers rely on the index of entity-labels provided by the knowledge graphs for generating candidates for entity disambiguation. For generating candidates over this index, EL uses string matching similarity. If an existing entity in the knowledge graph has a new alias or acronym which is not updated in the entity-labels index, these entities are not in the candidate list while searching through the new aliases, thus making the task of Entity Linking furthermore difficult.

- Inference: Certain entities are not directly available in the knowledge graph, but require some logical inferencing to identify them. Such entities are an instance of some parent entity having some specific properties with different name or slangs given to this entity. For example: "Barack Obama is an American politician." here the keyword 'American' will be linked to the entity 'United States of America' which is a country. The inference here is that person 'Barack Obama' is a 'citizen' of the country 'United States of America'.

2.3.2 Relation Linking

A KG consists of facts stored in terms of a triple. A triple represents a fact in terms of Subject Predicate Object. The process of annotating given natural language text with KG predicate is termed as Relation Linking. There are solutions to Relation Linking similar to Entity Linking that is, to first spot the span, generate candidate and then re-rank to get the correct annotation finally. Unlike Entity Linking, here finding the span for the keywords is not sufficient to generate the correct candidate space, as the expressivity of a predicate could vary more than that of named-entity.

Consider the following example of Relation Linking to get a better understanding
Example 1: question string: Who is the *wife* of Robert Downey Jr?
relation span: "wife".
correct predicate: `rel:spouse`

Example 2: question string: When was Robert Downey Jr *born*?
relation span: "born".
correct predicate: `rel:birthdate`

Example 3: question string: Where was Robert Downey Jr *born*?

relation span: "born".
correct predicate: `rel:birthplace`

In example 1, we can see that identifying the span for the relation could be sufficient for performing Relation Linking as the words "spouse" and "wife" are synonyms. Examples 2 and 3 showcases that the identified span for the relation is not sufficient as they leave much ambiguity. The span keywords could have a semantically different meaning, given the context from the whole question sentence.

Traditional pipeline based KGQA system have relied on Relation Linking to identify the KG predicates required to form the correct formal query corresponding to the natural language question. In various generic KGQA architecture such as OpenQA [31], FrankensteinQA [6] and OKBQA [32], Relation Linking has always been an individual component. The evaluation performed by Singh et al. [6] makes it evident that most QA systems suffer a dip of accuracy due to inaccuracy from Relation Linking. A more modern KGQA system merges the Relation Linking with the query building phase. Such advanced system provides the flexibility to the system as it can check for valid graph paths for constructing the formal query and also perform Relation Linking. Such approaches improve the accuracy of the KGQA system. Though within the scope of this section, we present Relation Linking as an individual component to get a deeper understanding.

Further the Relation Linking for knowledge graph question answering could be formalized as:

Formalization: Relation Linking

Given a natural language question Q_{NL}, assuming Knowledge Graph KG with collection of Relation $R = \{r_1, r_2, r_3, ..., r_n\}$, where n is a large number representing a total number of relations (r) in the KG. The Relation Linking system aims to identify all the relations in Q_{NL} and map to R .

Relation Linking needs to cover much more string variations as compared to entity labels. The following are the challenges that need to be overcome by a Relation Linking module.

- High semantic variations in Relation Linking: KG Relations can be expressed in natural language in multiple different ways using and not limited to synonyms, verb forms, noun forms, tenses etc. Thus there is a high lexical gap between the KG Relation labels and their occurrence in natural language.
- Multi-word and Multi-position A KG Relation may not be limited to a single word or adjacent words when expressed in the natural language. For instance "Where was RDJ born? " implies relation birth-place and "When was RDJ born?" implies relation birth-date.
- Identify Hidden Relations: Another challenge is to identify the relations which do not have a surface-form phrase in the natural language sentence. For example: "Name some Indian Astronauts?", here the keyword "Indian" refers to the hidden relation of "country of citizenship".
- KG Structure: For example, "Who is the brother of Justin Bieber?", a KG might have no separate property for ‘brother’ and ‘sister’ and would have a property for ‘siblings’. Thus, to solve this question, the formal query would require to fetch all siblings of the main-entity(Justin Bieber) and then select the male siblings from the retrieved results.

2.3.3 Understanding SPARQL

The content and factual information from the web are stored in Knowledge Graph using RDF, which is a directed, labelled graph data format. RDF is often used to represent personal data, social networks, digital metadata, and to provide an interface to a variety of information tools. SPARQL can be used for querying different data sources, whether the data is natively stored as RDF or viewed as RDF via middle-ware. SPARQL provides the ability to query the appropriate graph patterns and optional dial-up patterns. The query language syntax and semanticization of the SPARQL query language for RDF is defined by W3C specification [1] SPARQL supports even aggregation, subscriptions, rejection, expressions formation of values, extensible value checks and constraining source RDF graph queries. SPARQL query results may be set to results or RDF graphs.

SPARQL is W3C standard; it stands for SPARQL Protocol and RDF query language. The protocol part is essential for the developer who works on systems that pass sparql query back and forth between different machines. For most Question answering application, SPARQL's greatest value is to add a formal query language for RDF yet another w3c standard. SPARQL is an RDF query language that is, a semantic query language for databases able to retrieve and manipulate data stored in Resource Description Framework (RDF) format. It was made a standard by the RDF Data Access Working Group (DAWG) of the World Wide Web Consortium and is recognized as one of the key technologies of the semantic web.

The main building block of a SPARQL query is a *triple pattern*. Similar to an RDF triple, a triple pattern consists of placeholders, i.e., variables, at positions of a subject, predicate, or object (or all of them simultaneously). During the evaluation of a query, patterns are matched against the target RDF dataset so that variables turn into RDF terms that constitute a query solution mapping. *Basic Graph Pattern (BGP)* denotes conjunction of a set of triple patterns.

Triple Pattern, Basic Graph Pattern [24]

Definition 2.3.1 *Let $\mathcal{U}, \mathcal{B}, \mathcal{L}$ be disjoint infinite sets of URIs, blank nodes, and literals, respectively. Let V be a set of variables such that $V \cap (\mathcal{U} \cup \mathcal{B} \cup \mathcal{L}) = \emptyset$. A triple pattern tp is member of the set $(\mathcal{U} \cup V) \times (\mathcal{U} \cup V) \times (\mathcal{U} \cup \mathcal{L} \cup V)$. Let $tp_1, tp_2, \ldots, tp_n$ be triple patterns. A Basic Graph Pattern (BGP) B is the conjunction of triple patterns, i.e., $B = tp_1$ AND tp_2 AND $\ldots$ AND tp_n.*

Query forms: In the case of queries that read data from the database, the SPARQL language specifies four different query variations for various purposes.

SELECT query given the variables in the query, a Select query extracts the raw values from a SPARQL endpoint, and the results could be interpreted in tabular format.

CONSTRUCT query They are used to extract information from the SPARQL endpoint and transform the results into valid RDF.

ASK query They are used to provide a simple True/False result for a query on a SPARQL endpoint.

DESCRIBE query Used to extract an RDF graph from the SPARQL endpoint, the content of which is left to the endpoint to decide, based on what the maintainer deems as useful information. Each of these query forms takes a WHERE block to restrict the query, although, in the case of the DESCRIBE query, WHERE is optional.

[1]W3C Specification for SPARQL `https://www.w3.org/TR/sparql11-overview/`

SPARQL 1.1 specifies a language for updating the database with several new query forms.

SPARQL Endpoint

Definition 2.3.2 *A Web service that provides HTTP-based means for querying RDF datasets via the SPARQL protocol is denominated as a SPARQL endpoint. The endpoint implements a function f that maps from a query Q to solution mappings μ:*

$$f : Q \rightarrow \mu$$

2.3.4 Query Building - SPARQL

The goal of a KGQA is to retrieve answers from the knowledge graph for a given question. Most often, the answer fetched by the KGQA system is through hitting a formal query at the knowledge graph. Thus the problem of QA over KG becomes a translation problem such that the system aims to translate natural language questions to formal language query. Natural language is an understandable human form, whereas Formal language is machine-understandable. Thus the question asked by the human user has to be translated to machine understandable formal expression.

In most pipeline based QA systems Query Building is the last module which finally generates the formal query, SPARQL being a common formal query language. The entities and relations required for the query builder are provided by Entity linker and Relation linker, respectively.

Formalization: Query Builder

Given a natural language question Q_{NL}, with a set of identified and linked entities in the question as E, and a set of linked relation as R, the Query Builder QB uses this information to generate a formal query Q_{FL} which is semantically equivalent to Q_{NL}

For example, suppose we have the question Q_{NL} as “What is the capital of Germany“ where entity E is *ent:capital* and relation R is *rel:capital* , the query builder QB uses this information to generate query Q_{FL} as `Select ?ans where { ent:Germany rel:capital ?ans .}` , such that semantics of Q_{NL} are intact.

The other designs of a query builder do not take relations as inputs; instead, they perform Relation Linking while identifying the query paths. These Query Builders only need the question and entity Set for the input and produce the final output as the formal query. The advantage of such a query builder is that it does not propagates the error done during Relation Linking, which is one of the significant factors affecting the accuracy of a KGQA system. Over the years, we have seen researchers working in this core module of question answering pipeline and thus improving state of the art in the field.

Lets further look into more examples to get a better understanding of this sub-task in KGQA.

Example 1: Simple Question

Q_{NL} = Give all Robert Downey Jr. movie ?
E = ent:RobertDowneyJr.
R = rel:acted
Q_{FL} = `Select ?ans where { ent:RobertDowneyJr rel:acted ?ans}`

Example 2: Simple Question with class type
Q_{NL} = List all MCU movies with Robert Downey Jr. ?
E = ent:RobertDowneyJr., ent:MarvelStudio
R = rel:acted
Q_{FL} = `Select ?ans where {ent:RobertDowneyJr rel:acted ?ans . ent:MarvelStudio rel:producer ?ans }`

Example 3: Complex question
Q_{NL} = In which movie did RDJ played Iron Man
E = ent:RobertDowneyJr.
R = rel:parents
Q_{FL} = `Select ?ans where { ent:RobertDowneyJr rel:acted ?ans }`

In example 1, we have a simple question, so the SPARQL is easy to manage, as the template of the sparql is just one triple. Simplequestion dataset has; S P ?O. The second example shows the extension to the simple question; here we have class type involved; which in turn requires more effort from the QB. Third, is an example of a complex question where maintaining the semantics of the sparql query with respect to NLQ becomes a harder task. The relation and entity could be in multiple combinations, some of them could be rejected as they do not align with KG, but there are still a lot of candidates left. We could further have Count, Boolean, Ranking, filters and Union questions which could be easy to identify but very hard to form formal query.

2.4 Reified Knowledge Graphs

Knowledge Graph storing data in linked fashion often use RDF as the data format. While capturing meta-information such as qualifiers and references for an RDF triple, we need some way in RDF to describe the RDF triples themselves, which we refer to as Reification of Knowledge Graph [33]. Researchers have provided several solutions for Reification of KG such as n-ary relations, singleton properties, named graphs and more [34]. For our understanding, we will look into RDF* as it is a W3C standard and has gained popularity in the industry too.

RDF* using the basic principles of RDF, the RDF* is a extension to the standard RDF model [35, 36]. We, therefore, presume that three pairwise disjoint, countably infinite sets: $\mathcal{I}$ (IRIs), $\mathcal{B}$ (blank nodes), and $\mathcal{L}$ (literals). Thus, a tuple $(s, p, o) \in ((\mathcal{I} \cup \mathcal{B}) \times \mathcal{I} \times (\mathcal{I} \cup \mathcal{B} \cup \mathcal{L}))$ is an RDF triple and a set of RDF triples is called an RDF graph.

RDF* extends the notion of such triples by permitting a form of nesting; i.e., a triple may have another triple in its first or its third position. Such nesting may be arbitrarily deep. Formally, an RDF* triple is defined recursively as follows:

RDFstar [35]

Definition 2.4.1 *An RDF* triple is a 3-tuple such that*

- *Every RDF triple is an RDF$\star$ triple.*
- *Let t and $\tilde{t}$ be RDF$\star$ triples; for every $s \in (\mathcal{I} \cup \mathcal{B})$, $p \in \mathcal{I}$ and $o \in (\mathcal{I} \cup \mathcal{B} \cup \mathcal{L})$, the tuples (t, p, o), (s, p, t) and $(t, p, \tilde{t})$ are RDF* triples.*

Further adopting from [35], for the sake of conciseness, in this thesis we look into the syntax for SPARQL* that is based on Perez et al.'s [24] algebraic SPARQL syntax, and we focus only on the core concepts. For a more detailed formalization of SPARQL*, including the complete extension of the full W3C specification of SPARQL, please refer to the technical report [37] . We recall that the basic building block of SPARQL queries is a basic graph pattern (BGP); that is, a finite set of triple patterns, where every triple pattern is a tuple of the form $(s, p, o) \in ((\mathcal{V} \cup \mathcal{I} \cup \mathcal{L}) \times ((\mathcal{V} \cup \mathcal{I}) \times ((\mathcal{V} \cup \mathcal{I} \cup \mathcal{L})$ with $\mathcal{V}$ being a set of query variables that is disjoint from $\mathcal{I}$, $\mathcal{B}$, and $\mathcal{L}$, respectively. SPARQL* extends these patterns by adding the possibility to nest triple patterns within one another.

SPARQL*

Definition 2.4.2 *A triple* pattern is a 3-tuple that is defined recursively as follows:*

- *Any pattern $t \in ((\mathcal{V} \cup \mathcal{I} \cup \mathcal{L}) \times ((\mathcal{V} \cup \mathcal{I}) \times ((\mathcal{V} \cup \mathcal{I} \cup \mathcal{L})$ is a triple* pattern; and*
- *Given two triple* patterns tp and $t\tilde{p}$,and $s \in ((\mathcal{V} \cup \mathcal{I} \cup \mathcal{L})$, $p \in ((\mathcal{V} \cup \mathcal{I})$ and $p \in ((\mathcal{V} \cup \mathcal{I} \cup \mathcal{L})$, then (tp, p, o), (s, p, tp) and $(tp, p, t\tilde{p})$ are triple* patterns.*

2.5 Subdivision for Knowledge Graphs

We now do an introductory discussion of subdivision for Graphs and Knowledge Graphs, as it is a fundamental operation used in our EARL framework as discussed in Chapter 7.

In general, a subdivision of a graph G (also known as an expansion) is a graph resulting from the subdivision of edges in G. The subdivision of some edge e_{ij} with endpoints $\{v_i, v_j\}$ yields a graph containing one new vertex v_k, and with an edge set replacing e_{ij} by two new edges, $\{v_i, v_k\}$ and $\{v_k, v_j\}$

For example, the edge e_{ij}, with endpoints $\{v_i, v_j\}$, can be subdivided into two edges, e_{ij} and e_{ij}, connecting to a new vertex v_k, as shown in the Figure 2.5.

For a formal definition of Edge Subdivision and Graph Subdivision, let $G = (V, E)$ be a graph.

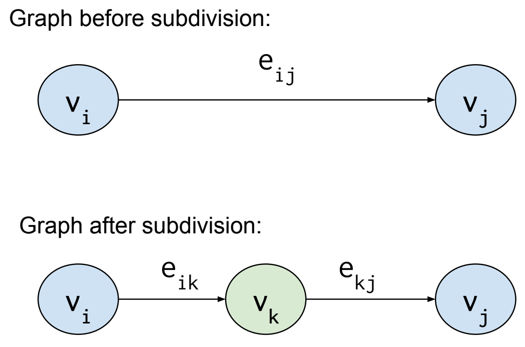


Figure 2.5: Basics of Subdivision Graphs

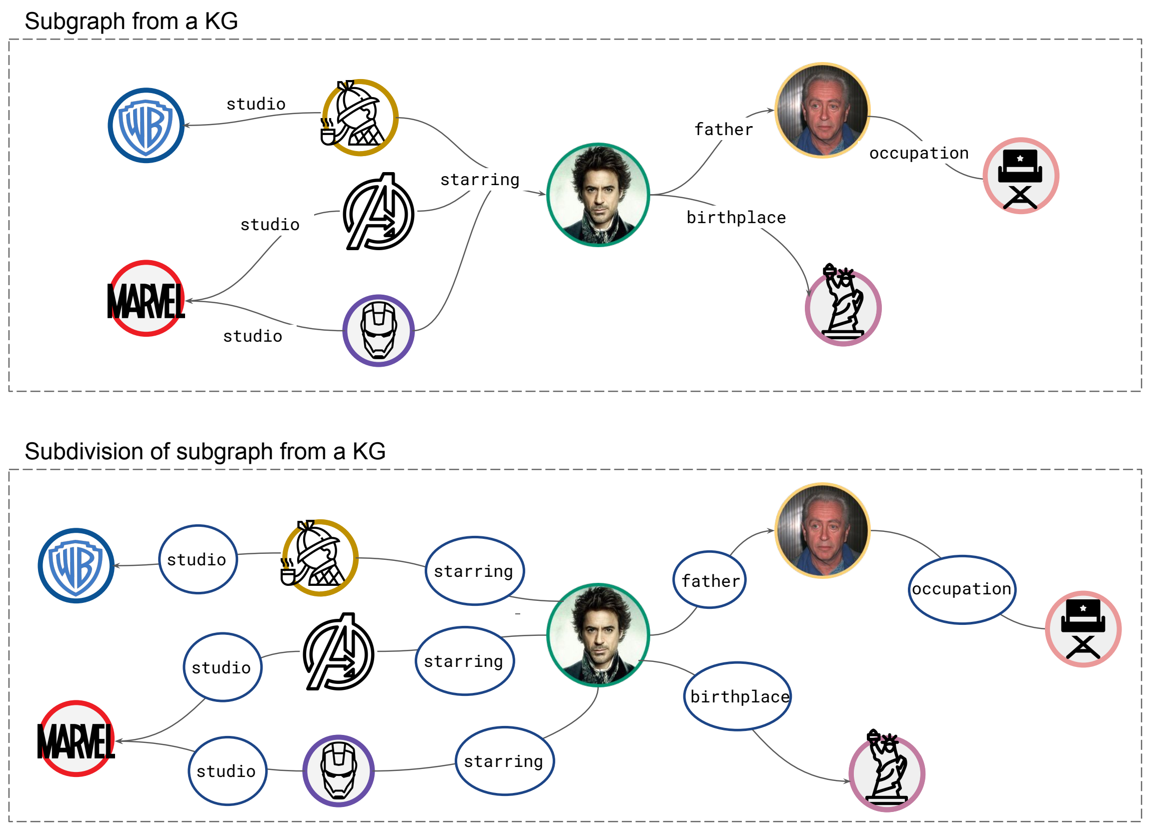


Figure 2.6: Application of Subdivision Graphs on Knowledge Graphs

Edge Subdivision

The edge subdivision operation for an edge $\{v_i, v_j\} \in E$ is the deletion of $\{v_i, v_j\}$ from G and the addition of two edges $\{v_i, v_k\}$ and $\{v_k, v_j\}$ along with the new vertex k

Graph Subdivision

A graph which has been derived from G by a sequence of edge subdivision operations is called a subdivision of G.

A Knowledge Graph is graph with entities on vertices and predicates on edges. A subdivision operation could also be performed on knowledge graph. In this special case, a predicate-edge is removed, and the addition of new predicate-vertex takes place. Thus the predicates will also move to the vertices. We have illustrated such changes in the KG in Figure 2.6.

2.6 Natural Language Generation (NLG)

Natural Language Generation is a study field since the 1960s and belongs to the domain of Natural Language Processing (NLP). NLG incorporates not only the fundamental aspects of Artificial Intelligence (AI) but also Cognitive Science (CS). The application such as text summarization, machine translation can be classified under the umbrella of language generation, we show an overview in fig 2.7. Our research focuses on the field of Question Answering, which is essential to natural language user interfaces[38] [39]. Within this thesis, we discuss the question-generation task in the related work section. In Chapter 8, we discuss answer-verbalization task in greater detail, which is a key contribution to the thesis.

Natural Language Generation (NLG) aims at generating automatic natural language text to meet specific user goals. This section largely contains materials from the key NLG book by Ehud Reiter and Robert Dale [40]. They define NLG as *“the sub-field of artificial intelligence and computational linguistics that is concerned with the construction of computer systems than can produce understandable texts in English or other human languages from some underlying non-linguistic representation of information“*. The input to an NLG system contains the Knowledge Source, which provides information about the domain, Communicative Goals, which describes the purpose of the generated text, User Models, which provides a characterization of the intended audience for whom the text is to be generated, and a Discourse History, which serves as a repository of information about previous interactions. As shown in Figure 2.8, NLG architecture contains three modules:
Document Planner: To determine the structure and content of a document.
Sentence Planner: To choose the syntactic structure and words to be used to describe the content and structure of a document. It also decides the ordering aspects.
Surface Realization: To map appropriate actual text to the microplanner’s abstract representation.

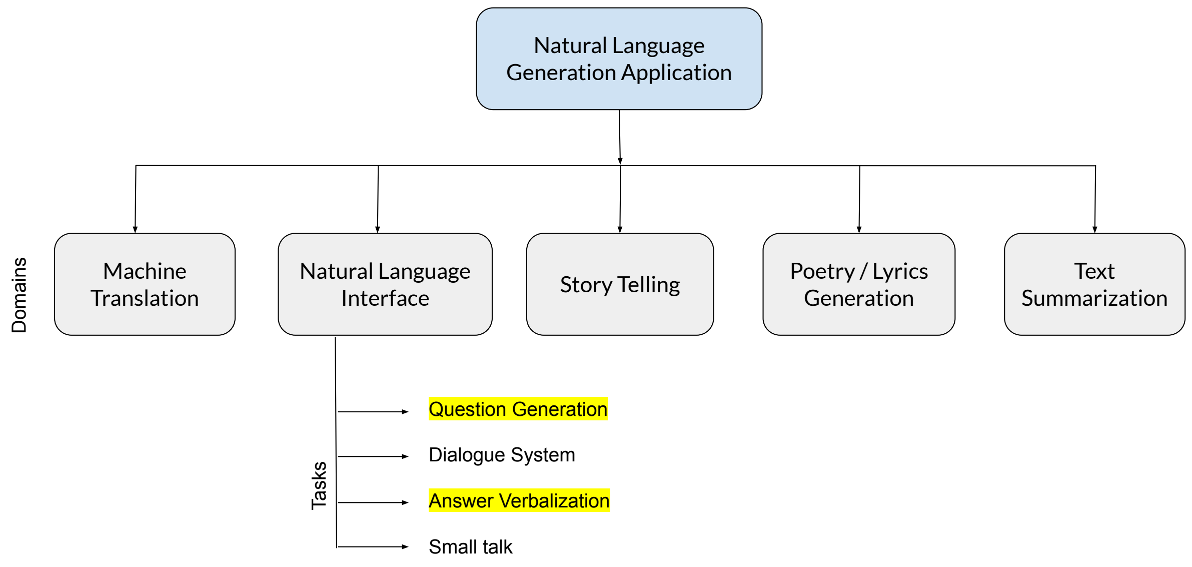


Figure 2.7: Application of NLG

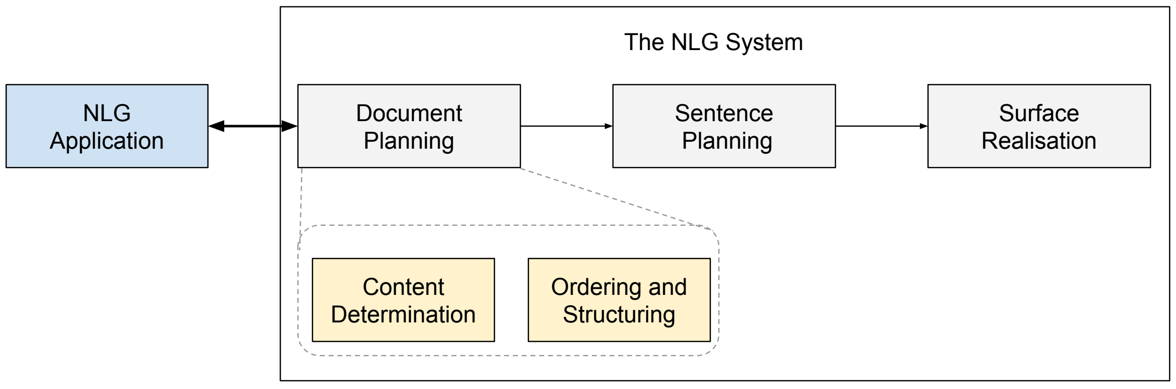


Figure 2.8: An overview of NLG Architecture

The different tasks associated with the content and structure of a document in the respective modules, as shown in Table 2.2, are discussed below:

- **Content Determination:** is accountable for deciding which content or information from the document is to be communicated in the generated text.
- **Document Structuring:** is accountable for deciding which portion or chunks of content from the document are to be grouped and how these different portions are linked linguistically.
- **Lexicalization:** is accountable for choosing the words or syntactic representation which is used to express the content selected by the content determination segment.
- **Referring Expression Generation:** is accountable for choosing the expression used to refer the entities in the document.

Module	Content Task	Structure Task
Document Planning	Content Determination	Document Structuring
Sentence planning	Lexicalization; Referring Expression Generation	Aggregation
Surface Realization	Linguistic Realization	Structure Realization

Table 2.2: Models and Tasks associated with NLG

- **Aggregation:** is accountable for deciding the mapping between chunks created by document planner and linguistic structures such as paragraphs or sentences.
- **Linguistic Realization:** is accountable for generating real natural language text from its abstract representation.
- **Structure Realization:** is accountable for generating mark-up symbols understandable by the document presentation segment from its abstract structures such as sections or paragraphs.

2.7 Sequence to Sequence (Seq2Seq) Models

In this section, we will discuss the most common NLP architecture, the Sequence-to-Sequence model or the Encoder-Decoder framework[41, 42][2]. These models are currently in use for several NLP tasks, e.g., machine translation, question answering, text summarization, speech recognition, etc., where the length of the input and output sequence are of variable length. The simple strategy behind this model is to use multi-layer RNNs such that: (i) the *encoder* RNN encodes the variable length input sequence into a fixed-length vector representation, and (ii) the *decoder* RNN conditions on this vector representation to generate another variable-length output sequence. The architecture of the Encoder-Decoder framework is shown in Figure 2.9.

In Mathematical terms: The goal of the model is to train the two RNNs jointly to maximize the conditional log likelihood for parameter set θ and a set of tuples $\{(X_1, Y_1), ..., (X_n, Y_n)\}$, where each pair (X_i, Y_i) consists of an input sequence $X_i = (x^{(1)}, ..., x^{(d)})$ and its corresponding output sequence $Y_i = (y^{(1)}, ...y^{(d')})$ with lengths d probably different from d':

$$\max_{\theta} \frac{1}{n} \sum_{i=1}^{n} \log p_\theta(Y_i|X_i) \tag{2.1}$$

Encoder Stage: In this step, the RNN sequentially reads each token of the input sequence x while changing its hidden state $h^{(t)}$ to generate a summary or context of the input sequence c, in accordance with the following equation:

$$\begin{aligned} h^{(t)} &= g(h^{(t)}, x^{(t)}) \\ c &= q(\{h^{(1)}, h^{(2)}, ..., h^{(d)}\}) \end{aligned} \tag{2.2}$$

where, g and q are non-linear activation functions.

Decoder Stage: The decoder is trained to predict the next token $y^{(t)}$, for a given hidden state $h^{(t)}$, to generate the final output sequence y. In this case, both the hidden state and the predicted token depend on

[2]Note that the only difference between the two approaches, [41] and [42], is that the former uses LSTM in place of RNN as the Encoder and Decoder

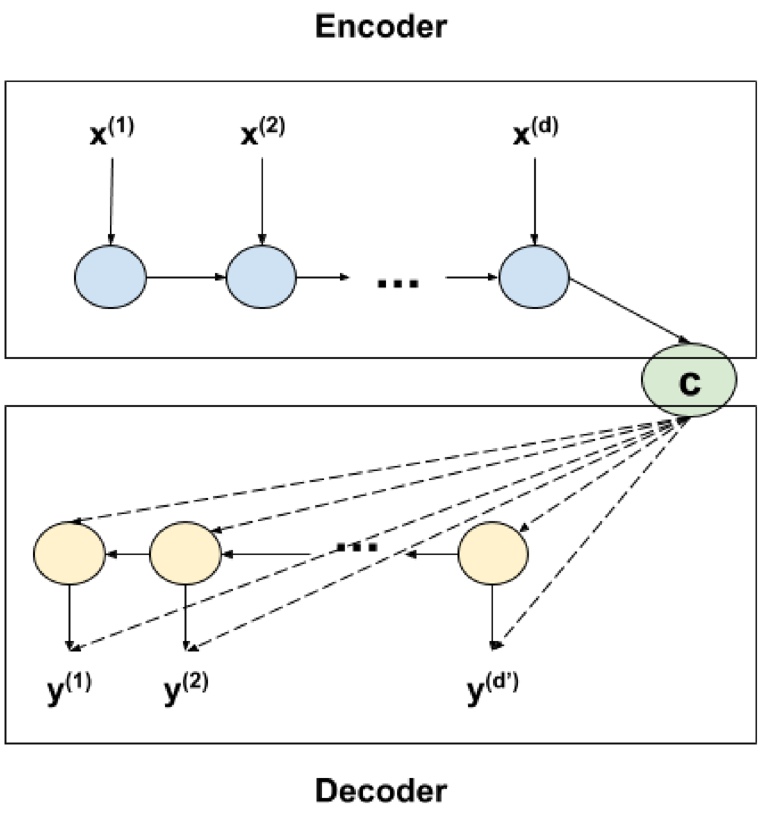


Figure 2.9: Encoder Decoder Framework

previously predicted token $y^{(t-1)}$ and on the context vector c. The probability distribution of the predicted token is given by:

$$\begin{gathered} h^{(t)} = g(h^{(t-1)}, y^{(t-1)}, c) \\ p(y^{(t)}|y^{(t-1)}, y^{(t-2)}, ..., y^{(1)}, c) = f(h^{(t)}, y^{(t-1)}, c) \end{gathered} \tag{2.3}$$

where, f is an activation function producing valid probabilities, such as a softmax function. The probability over the output sequence y can be viewed as a decomposition of joint probabilities into ordered conditionals as follows:

$$p(y) = \prod_{t=1}^{d} p(y^{(t)}|y^{(t-1)}, y^{(t-2)}, ..., y^{(1)}, c) \tag{2.4}$$

The sequence-to-sequence mechanism, stated in [43], is an updated version of this model with an attention mechanism implemented in the decoder stage. Another variation implements CNN in place of RNN.

2.8 Word Embeddings

Word Embeddings are a real-valued vector representation of words, based on a predefined fixed size vocabulary, depicting their semantic meaning in the context of the training corpus. This helps in answering analogy questions as well as reasons out the meaning of words using vector algebra.

A word analogy is a statement of the form "a is to b as p is to q", claiming that a and p are transformed in the same manner as b and q. This can be expressed in terms of conditional probability for a word w as follows:

$$\frac{P(w|a)}{P(w|b)} = \frac{P(w|p)}{P(w|q)} \tag{2.5}$$

and in terms of word vectors:

$$\begin{aligned} \vec{v_w} \cdot \vec{v_a} - \vec{v_w} \cdot \vec{v_b} &= \vec{v_w} \cdot \vec{v_p} - \vec{v_w} \cdot \vec{v_q} \\ \text{or,} \quad \vec{v_w} \cdot (\vec{v_a} - \vec{v_b} - \vec{v_p} + \vec{v_q}) &= 0 \end{aligned} \tag{2.6}$$

For a better understanding of analogy, we consider example "*king is to man as queen is to woman*" (see Figure 2.10). Assuming the probabilities of words are factorized with respect to different dimensions of the word as follows:

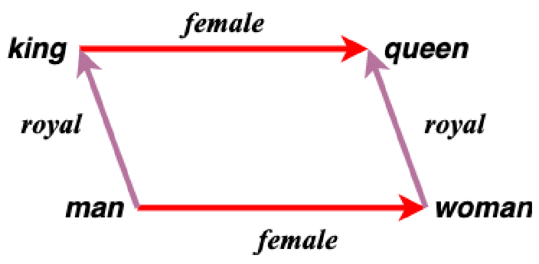


Figure 2.10: Example of Word Analogy used Word Vectors

$$\begin{aligned} P(w|king) &= f(w|gender = male) \times f(w|status = royal) \times P(w|is\_a\_human) \\ P(w|man) &= f(w|gender = male) \times f(w|status = normal) \times P(w|is\_a\_human) \\ P(w|queen) &= f(w|gender = female) \times f(w|status = royal) \times P(w|is\_a\_human) \\ P(w|woman) &= f(w|gender = female) \times f(w|status = normal) \times P(w|is\_a\_human) \end{aligned} \tag{2.7}$$

Hence,

$$\frac{P(w|king)}{P(w|man)} = \frac{f(w|status = royal)}{f(w|status = normal)} = \frac{P(w|queen)}{P(w|woman)} \tag{2.8}$$

Computing embeddings require a substantial amount of resources, but fortunately a wide range of pre-trained representations are open-sourced and can be used for most application. Some of these pretrained embeddings models are:

- *Word2Vec* [44] is pretrained embedding model by Google trained on English News articles.
- *GloVe* or Global vectors [45], developed at Stanford, is an extension of the *Word2Vec* model.
- Facebook's *fasttext* [46] is multi-lingual (294 languages) vector representation utilizing subword information.

CHAPTER 3

Related Work

A review of existing literature is necessary to establish the basis for new work impart a direction to new research. Knowledge of salient points and limitations of the current state of research further highlights the research gap and scope for improvement. This section highlights some of the prominent existing approaches pertaining to the various aspects of KGQA. We first study some fundamental works about Knowledge Graphs and their evolution. We then look into various approaches in use by current question answering systems. We then proceed to study various techniques used in entity linking and relation linking. At the end, we discuss existing techniques for natural language generation targeted towards answer verbalization.

3.1 Knowledge Graph

Knowledge Graphs(KG) are a repository of knowledge about worldly entities. By the classical definition of Knowledge Graphs, entities are nodes and properties are edges of the graph. KGs store various kinds of information for entities, and this information is interlinked, generating a web of enriched knowledge for a particular genre or field. Let us suppose we have an entity, "Barack Obama", in the knowledge graph. This node of the KG may have properties like birthplace, spouse, office and other information. The same entity-node may have a connection to the KG-entity "President of United States of America", to which all other former and present presidents would be connected. All presidents would have similar property information as Barack Obama. For the users, it becomes possible to retrieve information such as "get all the presidents of the USA who were born in New York".

Google pitched the term knowledge graph as they started using the Google knowledge graph in their search queries. Initially, they displayed interlinked information in the UI (user interface) [1] Though similar information storage was also used previously under the name of knowledge bases, the term Knowledge Graph has superseded, it is more apt since the underlying technique is a graph.

The knowledge graph technology has attained significant success, and large organizations and product-oriented companies such as eBay, Amazon, and others have adopted KGs into production. The most popular general purpose KGs are DBpedia, Freebase, Yago, Wikidata, Google-KG. Other than the general-purpose knowledge graphs, we also see domain-specific KGs. For example, for scholarly communication, we have KG such as ScholarlyData [47] and Microsoft Academic Graph [48]. MusicBrainz [49] knowledge graph is an open and public music encyclopedia that collects music metadata. Knowledge Graphs are also used in tourism and cultural heritage. BabelNet [50] is a multilingual semantic network

[1] refer to YouTube video. `https://www.youtube.com/watch?v=mmQl6VGvX-c`

Parameters	Wikidata	DBpedia	Freebase	Yago
Number of triples	748 530 833	411 885 960	3 124 791 156	1 001 461 792
Number of classes	302 280	736	53 092	569 751
Number of relations	4839	2819	70 902	88 736
Number of entities	18 697 897	4 298 433	49 947 799	5 130 031
Avg. number of entities per class	61.9	5840.3	940.8	9.0

Table 3.1: A Statistical Comparison of Knowledge Graphs, Adopted from Färber et al [8](2018).

and encyclopedic dictionary. In the scope of this thesis, we only look into the general-purpose KGs which are available publicly.

Semantic Web community's need for a new model of structured information representation and management led to the establishment of DBpedia [51, 52]. DBpedia is based on the RDF data model and mapped to DBpedia ontology. The goal of DBpedia is to provide a large scale rich corpus of diverse data, enabling the developer community to evaluate various techniques and standards of the semantic web. In the year 2007, Wikipedia was one of the most read websites by the users of the internet and received constant enrichment in information by wiki-editors. By design, DBpedia seeks to draw information from Wikipedia into a large scale RDF corpus. The DBpedia project organizes the Wikipedia content such as info-box information into structured knowledge, allowing Semantic Web techniques to explore this knowledge.

Freebase [53] was a comprehensive distributed knowledge base consisting mainly of the data of closed community workers. It was an online collection of organized data from several sources, including individual wiki contributions submitted by users. Freebase aimed at providing a global platform that enabled more efficient access by individuals (and machines) to widespread knowledge. The company Metaweb founded Freebase and ran publicly since March 2007, and later got acquired by Google in July 2010. Freebase is a convenient and scalable database that is used to organize human intelligence in general. The data is developed, organized and maintained in Freebase collaboratively. Freebase uses a proprietary graph model for storing complex statements. Freebase shut down its services on June 2015. Later Freebase data was integrated into Wikidata.

Freebase extensively contributed to various KG related research areas, but with its discontinuation, the community will move to other KGs eventually. We have already observed the migration of KGQA datasets from Freebase to other KGs [54]. From a KGQA perspective, Wikidata and DBpedia are the two most important KGs as we move forward. A significant advantage in DBpedia is the hierarchy in classes, as it uses a defined ontology, which could be exploited by KGQA systems. On the other hand, Wikidata has a flatter hierarchy in classes; thus, we can see a large number of classes in Table 3.1. However, Wikidata contains more information as compared to other KGs and receives updates continuously by the wikidata-editors. Hence, for KGQA, information is always kept updated so that one gets recent results in response to the query.

3.2 Question Answering

Based on the survey by Hoffner et al. [55] Semantic Question Answering (SQA) is characterized by

1. User posing questions in natural language (NL)

2. User not being bound to any terminology, and free to choose his/her own words
3. User receiving a concise answer to the input question from the knowledge graph

The goal of KGQA is for the user to be able to ask questions in a free-flowing language. It is not supposed to be required for the user to obtain the skills of formal query languages such as SPARQL, nor worry about the vocabulary used in the knowledge graph. Natural language has a complex grammar, and it is hard to interpret all its semantics for a machine. Sometimes, the language is ambiguous even to humans. For some standard NLP tasks such as POS-tagging(parts of speech), parsing, etc., we have mature components. However, techniques required for question answering are still not sophisticated and face several challenges, as discussed in the background section.

We classify question answering into two primary strategies

1. Semantic Parsing based systems, which rely on a multi-component pipeline architecture.
2. Towards End to End question answering using machine learning and deep learning.

3.2.1 Question Answering - Semantic Parsing

Over the last decade, several *KGQA* approaches attempt to translate Natural Language Question (NLQ) into Formal Query like SPARQL. Early works in this direction include GiNSENG [56]. It is a guided input NL search engine, that does not understand NL queries, but uses menus to formulate NL queries in small and specific domains and allow users to query OWL knowledge bases in a controlled language akin to English. Subsequently, Semantic Crystal [57] was proposed, which is also a guided and controlled graphical query language. Systems such as AquaLog [58] and its advancement, PowerAqua [59], are based on linguistic mapping structures to ontology-compliant semantic triples. PowerAqua is the first system to perform QA over structured data. It provides a single NL query interface for integrating information from heterogeneous resources. The limitation of PowerAqua is the lack of support for query aggregation functions. Along the same lines, FREyA [60] allows users to enter queries in any form and uses ontology reasoning to learn more generic rules. It also provides for better handling of ambiguities over heterogeneous domains. But FREyA requires some level of effort in KB structure understanding to clarify disambiguation efficiently. Also, it highly depends upon modeling and vocabulary of the data at the user-end, making it inadequate for a naive user. Other works such as NLPReduce [61] allow users to pose questions in full or slightly controlled English. NLPReduce is a domain-independent system, which leverages the lexico-syntactic pattern structures of query input to find better matches in the KB. It maps query tokens with synonym enhanced triple stores in the target corpus, based on which it generates SPARQL statements for those matches. QTL [62] is a feedback mechanism for question answering using supervised machine learning on SPARQL. TBSL [63] uses BOA patterns as well as string similarities to fill the missing URIs in query templates and bridge the lexical gap.

Over the years, Question Answering over Linked Data (QALD) has become a popular benchmark. In QALD-3 [64], SQUALL2SPARQL [65] achieved the highest precision in the QA track. SQUALL2SPARQL takes an input query in SQUALL, which is a special English based language and translates it to SPARQL. Since no linguistic resource is required, it results in high performance. But on the other hand, it makes the SQUALL query unnatural to the end-user and requires manual annotations of the URIs. In QALD-4 [66], GFMed [67] achieved the highest precision in the biomedical track. It is based on the Grammatical Framework (GF) [68] and a Description Logic-based methodology. It proposes an algorithm to translate a query in natural language into SPARQL queries using GF resources. It can support complex queries but only works with controlled languages and biomedical datasets. In

POMELO [69], predicates of the RDF triples are mapped to frame predicates while the subjects and objects are mapped to core frame elements. Then after a question abstraction step, the final SPARQL query is generated. POMELO is based on a closed environment (biomedical) and fails to relate the disconnected semantic entities. gAnswer [70] proposes a graph mining algorithm to map natural language phrases to top-k possible predicates to form a paraphrase dictionary. It also proposes a novel approach to perform disambiguation in the query evaluation phase, which improves the precision and speeds up the query processing time greatly.

Xser [71], the most successful system in QALD-4 and QALD-5, uses a two-step architecture. It first understands the NL query by extracting phrases and labeling them as *resource, relation, type* or *variable* to produce a Directed Acyclic Graph (DAG). This semantic parser works independently of any Knowledge Base (KB), and later these semantic entities are instantiated with the given KB. However, it requires excessive human involvement in manually annotating the questions with phrase dependency DAG. Xser uses semantic parser with DAG as a linguistic analyzer. Further, Xser uses wikipedia miner tool to generate the candidate set of DBpedia entities, and PATTY to map phrases to predicates and categories of DBpedia.

APEQ [72], from QALD-5 [72], uses a graph traversal based approach, where it first extracts the primary entity from the query and then tries to find its relations with the other entities using the given KB. APEQ uses Graph traversal technique to determine the main entity by graph exploration. Finally, the graph with the best scoring entities is returned as the answer.

As understanding natural language is a difficult task on its own, other researchers have proposed the use of controlled natural language (CNL) [73].A CNL system adds constraints to the grammar of user input language such that, (i) the language is 'formal', hence easily interpretable by machines, and (ii) the language is still 'natural' enough to cover many semantic variations posed by the user. CANaLI (an acronym for Context-Aware controlled Natural Language Interface) is a CNL based QA system. In QALD 6, CANaLI was at the top of the leader-board. CANaLI controls and helps the users in typing input questions. It allows user to give input question only that are semantically correct concerning the underlying KG, and syntactically coherent to the grammar of its CNL. If the user hesitates in typing the input question, the CANaLI system starts suggesting correct completions. Thus the user can learn to use CANaLI easily and quickly.

Diefenbach et al. [74] proposed a multilingual, KG agnostic QA system[2]. This system only considers the semantics of words in the question and ignores the syntax of the question. For example, given the question 'List the authors born in Germany', the system uses the semantic relevant words 'authors','born' and 'Germany'. Now, even with a paraphrased question as 'Give me Germany born authors' or just keywords 'Authors, born, Germany', the system will perform the same. As shown in Figure 3.1, this system follows four stages in its pipeline. In the expansion stage, the system removes stop words and perform n-gram based search to retrieve all possible IRIs from an index of IRI and labels. From these IRI set, construct SPARQL by fitting to predefined sparql templates in Query Construction stage. The goal here is to remove all such sparql candidates which returns a set as a result. Here the system only considers four graph patterns for single-triple and two-triple SPARQL. Once we have the cleaner set of SPARQL candidates, the system re-ranks candidates based on several features such as word coverage by sparql, edit distance. In the last stage, logistic regression supports the Answer Decision.

[2](demo for QAnswer:`https://qanswer-frontend.univ-st-etienne.fr`)

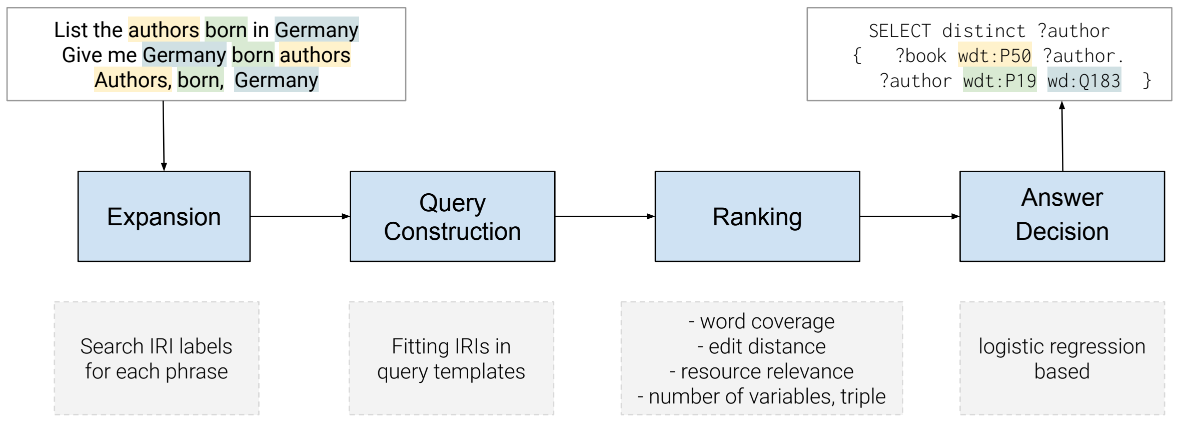

List the authors born in Germany
Give me Germany born authors
Authors, born, Germany
SELECT distinct ?author
{ ?book wdt:P50 ?author.
?author wdt:P19 wd:Q183 }
Expansion
Query Construction
Ranking
Answer Decision
Search IRI labels for each phrase
Fitting IRIs in query templates
- word coverage
- edit distance
- resource relevance
- number of variables, triple
logistic regression based

Figure 3.1: Pipeline of WDAqua Core-1. (The final sparql is corresponding to Wikidata)

3.2.2 Question Answering - Machine Learning

The recent survey by Chakraborty & Lukovnikov et al. [75] provides a comprehensive introduction to KGQA approaches, based on neural networks and machine learning. Here, the KGQA problem is defined as a semantic parsing problem, where the goal is to train the model that captures the semantics of the question, and retrieve the answer from the underlying KG. They divide the prediction models commonly used in semantic parsing into three main categories: (i) classification (ii) ranking and (iii) translation.

In a simple question scenario, there is a fixed query structure to map to just one query pattern. Example: the question "Who is the wife of Robert Downey Jr?" will have a single triple query "ent:RobertDowneyJr rel:spouse ?ans". Here QA system only needs to find the entity and one relation in the question to get the answer, and there is no structural variation from the query perspective. With these restrictions, simple text classification could predict the correct entity and relation. Entity span detection has been treated as a classification problem, such that given a token, the classifier generates a response indicating whether the token is in entity span or not [76].

Lukovnikov et al. [77] developed an end-to-end neural network approach for KGQA, restricting to the simple questions. They develop the entire solution into a single process rather than component-based architecture, overcoming propagation errors commonly found in pipeline based architectures. The proposed approach uses context from both character and word level embeddings, for both the entity and relations, as shown in Figure 3.2. By using character level context, it handles out of vocabulary entity words and at the same time, it has the advantage of word-level semantics. The model is relatively simple as it does not adopt any attention mechanism or separate segmentation models, yet it advances state of the art in the simple question KGQA. For a different context, we could reuse such solutions by retraining the model.

When we relax the constraints of a single formal query template and move towards the complex questions, the number of ways a query could get structured increases exponentially. In such a case, it is no longer feasible for the QA system to rely only on a classifier approach, and this is where the ranking and translation approach has been more popular.

In a ranking approach, given a question, we try to generate many possible formal queries (paths) around the subgraph of the primary entity identified in the question. The task is to re-rank the list of queries based on the semantic interpretation of the question and queries so that the highest-ranked query retrieves the answer to the given question.

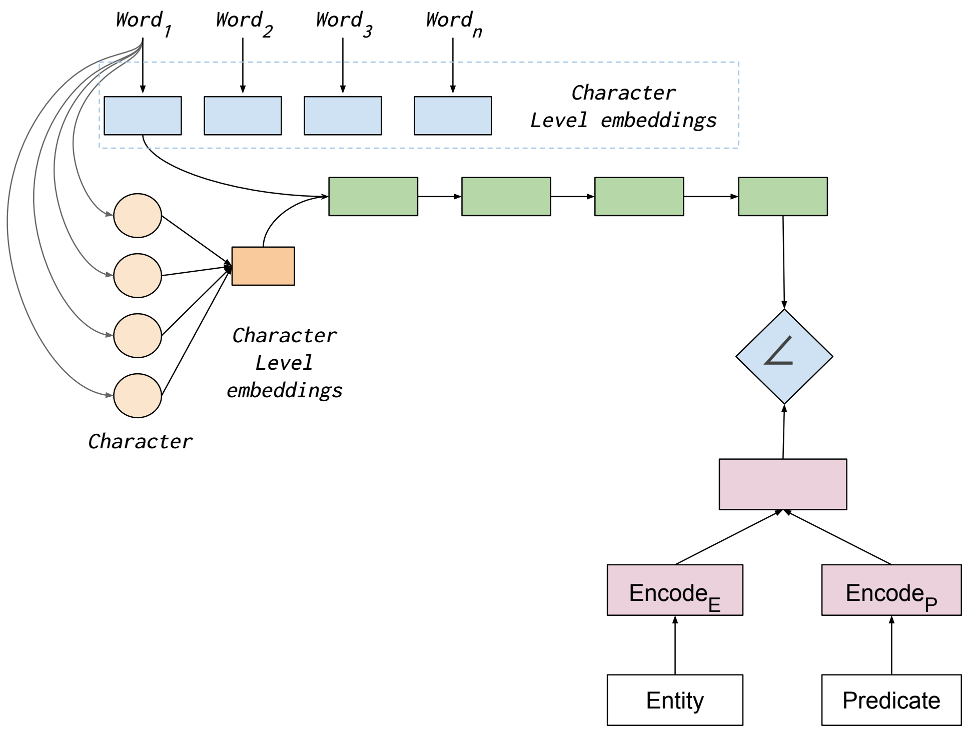


Figure 3.2: Using Word and Character Level Embedding to Solve Simple Question Answering [77]

Yih et al. [78] (STAAGG) uses query graph for semantic parsing of a given question. Query graphs are closely related to L-DCS and L-calculus. STAGG follows three steps: i) Identify the topic entity of the question, ii) Identify the relationship between the topic entity and the answer ii) Expand the query graph such that it covers all the constraints of the subgraph which the answer needs for a correct interpretation. Here they coin the term "core-chains", which are the generated paths around the topic entity to answer. Once the list of core chains are generated, and all the features have been extracted, STAGG treats to solve the QA problem by re-ranking the core-chains and by expanding the highest-ranked core-chain to a formal query to retrieve the answer.

Further Maheshwari et al. [5] performed an empirical study over the query graph ranking approaches used in KGQA. In this work, they investigate to find models more suitable for the KGQA task besides providing insight into the training settings and limitation to these model. They advance the state of the art in this research area by the proposed novel slot matching model. The proposed model can exploit the query graph structure by comparing the different representation of the question.

Tackling KGQA as a translation problem concentrates on the methods by which a sequence of the tokens form the logical expression rather than choosing the appropriate logical expression among several previously generated candidates. In this setup, KGQA is modelled as a translation problem, where we need to translate an NLQ Q_{NL} into a formal query Q_{FL} that can be expected to return the intended answer when executed over the source KG K.

Golub & He [79] proposed a character-level encoder-decoder framework that significantly improves performance over state-of-the-art word-level neural models while providing a much more compact model

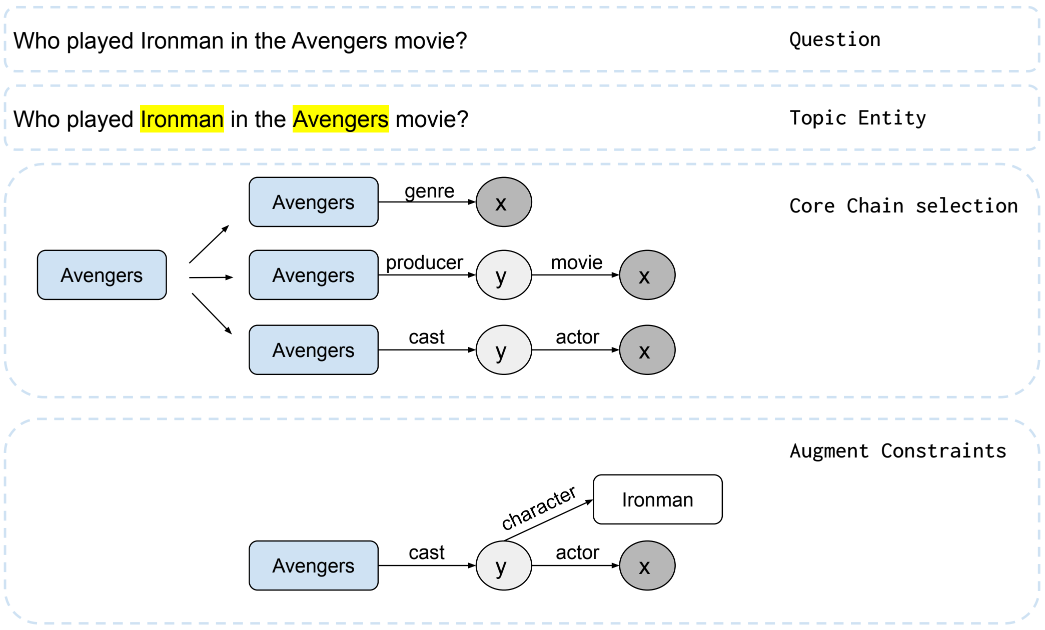


Figure 3.3: Showcasing Core Chain Approach

that can be learned from fewer data. Guo et al. [80] regard the generation of a logical form as the prediction of a sequence of actions, each of which corresponds to a derivation rule in the flexible and straightforward grammar. Here they introduced a generative model that interprets the logical form of an utterance in a top-down manner. They developed a grammar-guided decoder to generate possible action sequences following the grammar.

3.3 KGQA Datasets

In the past, KGQA community has developed several datasets for improving the state of the art of the question answering system, so systems may set a benchmark. For the Freebase Knowledge Graph, we had free917 [81] dataset, where natural language questions have corresponding lambda calculus as formal expression. These questions are from 81 Freebase domains, covering over 600 different relations.

In 2015 Bordes [82] proposed a large dataset of 100k questions with simple sparql formation. In this dataset, each question is limited to a single triple pattern (S P O). The natural language question expresses Subject and Predicate, and the answer was the Object of the triple. The collection process of SimpleQuestions has a two-step process. First, curating single triples from Freebase for question annotation. Filtering is done based on the handcrafted parameters, to remove noise from the dataset, and make meaningful questions. The second step is to provide human annotators with facts(single triple) and assign the task to frame a question based on the Subject and Predicate. The human annotators have a choice to use freebase.com so that they could gain background knowledge about the given subject entity to frame a better question. Also, the human annotator had the choice to skip the instance of the task.

Further, Serban & Garćıa-Durán [11] extended SimpleQuestion dataset from 100k question to 30B questions using the recurrent neural network. They modeled the problem of question generation to a neural machine translation task, where given a triple (formal language) the task is to generate a question (human language), while intentionally leaving out the *object*. Given a set of <*subject predicate object*>

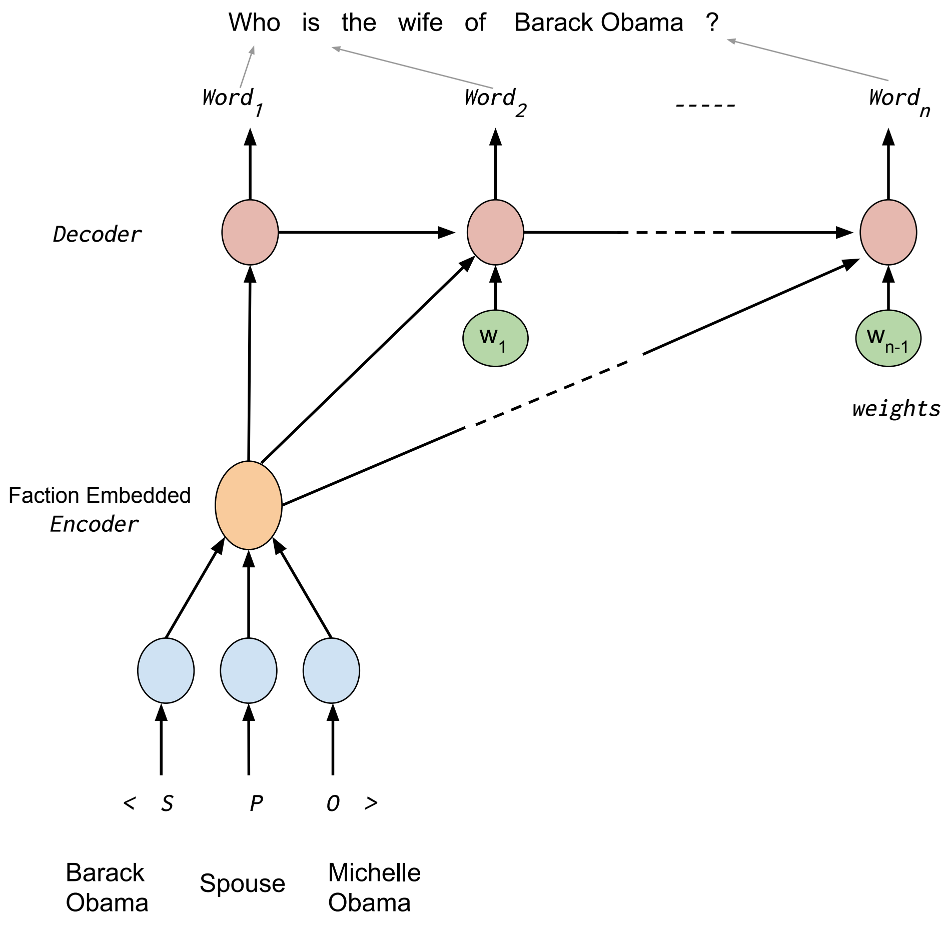


Figure 3.4: Simple Question Generation using Recurrent Neural Network

in form of a triple, the embeddings of the factoid is encoded by the encoder. The decoder reads fixed-size vectors and generates the output sequence as a natural language question.

Another popular complex question KGQA dataset over Freebase is WebQuestions [14]. This dataset was curated using Google suggest API, by starting 'wh' question and entity name, collecting 1M, such suggested-questions, and then these questions were hosted in Mturk experiment. The task for the Mturks was to get the answer to these question only using Freebase. The final dataset turned out to be 6642 questions answerable on Freebase. Later, this dataset was extended as WebQuestionSP by annotating with the formal expression to the questions.

The research community doing question answering over DBpedia started QALD challenge in 2013. Here they proposed several tasks for question answering such as Multi-lingual question answering, Hybrid Question Answering, QA in Medical Domain and others. The QALD Multi-lingual QA task gathered the most attention by the research community. In this challenge, the data set of approx 100 questions were curated by the field experts and test was setup. Every next challenge question from the previous challenge was released as the training set, and a new set of 50-100 question was released as a test set. The QALD tasks have high quality and wide variety as the field experts handcraft the limited number of the questions.

Data Set	Size	Variation	Formal Language	Target KG
Simple Questions[10]	100K	low	SPARQL	Freebase
30M Factoid Question[11]	30M	low	SPARQL	Freebase
QALD-9[13]	563	high	SPARQL	DBpedia
Free917[81]	917	medium	λ-Calculus	Freebase
WebQuestionSP[12]	5k	medium	SPARQL	Freebase
ComplexWebQuestionSP[83]	34K	medium	SPARQL	Freebase

Table 3.2: A Comparison of KGQA datasets

Data Set	Curating procedure
SimpleQuestions[10]	A factoid triple is shown to the user to pose a NL Question with Object being the answer
30M Factoid Question[11]	Using recurrent neural network to learn on SimpleQuestions and expand the dataset.
QALD-9[13]	Experts from the QA field design a small set of question with high quality and variety.
WebQuestions[14]	Question curated from Google Suggest API, and then identify the questions answerable using Freebase.
WebQuestionSP[12]	Extending WebQuestions with their corresponding formal expression .
ComplexWebQuestion[83]	Further adding complexity and constraints to SPARQL query of [12] and thus making complex questions

Table 3.3: A Comparison of Curating Procedure of KGQA Datasets

3.4 Linking Text to Knowledge Graph

3.4.1 Entity Linking

Entity Linking is the process of connecting the entities in the text to a knowledge graph or knowledge base. For this purpose, the first step is to find the entity mentions in the free text; known as entity recognition. The second step followed is called entity disambiguation, where the process of linking the identified entity to knowledge base occurs.

One of the basic approaches is NER for Knowledge-based systems, where all the possible entity tokens are known beforehand. These systems could be used in a domain-specific use-case as the lexicons for the entities could be limited. One significant disadvantage here is that these systems can not identify a new entity in the text. Some of the earlier approaches of NER are using bootstrapped unsupervised methods. These systems are based on grammar rules and feature engineering such as extracting patterns over annotated data.

Collobert and Weston (2008) [84] proposed one of the first neural network architectures for NER, with feature vectors constructed from orthographic features (e.g., capitalization of the first character), dictionaries and lexicons. Further work replaced these manually constructed feature vectors with word embeddings (Collobert et al., 2011) [85], which are representations of words in n-dimensional space, typically learned over large collections of unlabeled data through an unsupervised process such as the skip-gram model (Mikolov et al., 2013) [86].

The entity linking challenge has attracted a wide variety of solutions over time, with more attention to the entity disambiguation phase. Linking natural language phrases to DBpedia resources, Spotlight [18] breaks down the process of entity spotting into four phases. First, it goes through spotting phase using the Aho-Corasick string matching algorithm. Then it identifies the entity using a list of surface forms and then generates DBpedia resource candidates. It then disambiguates the entity based on surrounding context such as a paragraph. This system goes beyond the standard TF-IDF model and introduces the Inverse Candidate Frequency (ICF) weight. Lastly, DbpediaSpotlight provides flexibility through its Configuration feature, where the user could set parameters such as Resource to annotate, Resource prominence, Contextual Ambiguity, Disambiguation Confidence. AGDISTIS [19] follows the inherent structure of the target knowledge base more closely to solve the problem. Being a graph-based disambiguation system, AGDISTIS performs disambiguation based on the hop-distance between the candidates for the entities in a given text, where multiple entities are present. Babelfy [87] uses word sense disambiguation for entity linking. On the other hand, S-MART [88] is often appropriated as an entity linking system over Freebase resources. It generates multiple regression trees and then applies sophisticated structured prediction techniques to link entities to resources. Falcon[89] performs entity and relation linking, which being a heuristics-based system, does not involve machine learning. Their motto being "Old is Gold", they build a rule-based system where they rely on dataset-specific morphological rules. As with any rule-based system, they perform stronger on certain datasets while much weaker on the datasets they were not hand-tuned on. EARL(which we discuss later in Chapter 7) on the other hand delivers more consistent performance across all datasets [90].

In the early 2010s, end-to-end models which combined entity span detection and entity disambiguation in one model started with attempts to build feedback mechanisms from one step to the other so that the next stage can recover error in the prior stage. One of the first attempts, Sil et al. [91] uses a popular NER model to generate the extra number of spans and let the linking step take the final decisions. Their method, however, depends on a good mention spotter and uses hand-engineered features. Later, Luo et al. [92] developed competitive joint MD and ED models using semi-Conditional Random Fields (semi-CRF). However, the basis for dependency was not robust, using only type-category correlation features. The other engineered features used in their model are either NER or ED specific. Moreover, although their probabilistic graphical model allows for low complexity learning and inference, it suffers from high computational complexity caused by the usage of the cross product of all possible document spans, NER categories and entity assignments. Another solution that J-NERD [93] addresses are the end-to-end task using engineered features and a probabilistic graphical model on top of sentence parse trees. More recently, Kolitsas et al. [94] worked on a genuinely end-to-end MD (Mention Detection), and ED (Entity Disambiguation) combined into a single EL (Entity Linking) model. They use context-aware mention embeddings, entity embeddings and a probabilistic mention - entity map, without demanding other engineered features.

Recently, there has been some work on entity linking for short text on Wikidata[95]. Opentapioca [96] works on less number of classes but is openly available both as a demo and as code, and is lightweight. Falcon 2.0 [97] is a rule-based EL solution on Wikidata which is openly available and fast, but it does not scale nicely to new datasets without manual rule additions. While Sevigli et al. [98] performs ED using KG entity embeddings (DeepWalk [99]) on Wikidata, Sorokin et al. [100] not only uses KG entity embeddings (TransE [101]), but also performs MD and ED end-to-end in a single model. On the other hand, KBPearl [90] is more recent work on KG population which also targets entity linking as a task for Wikidata. It uses dense sub-graphs formed across the document text to link entities.

3.4.2 Relation Linking

Relation linking is connecting phrases in natural language to ‘predicates‘ present in the Knowledge Graph. To achieve this task, the research communities have worked on several approaches, and we can broadly classify them in two basic categories. First is to have an extensive database where these mapping of natural language phrase and predicate labels are already present. The second approach is to be able to train a machine learning model to learn these semantic similarities.

A primitive way for relation linking is string similarity between text and predicate, but natural language is versatile since a large number of words and phrases could be used to express a similar property in the text. We can see the use of the standard dictionary (for example Oxford Dictionary) to expand the vocabulary of the string matching algorithms by making use of the synonyms. Thus, one can start creating an extensive database using these techniques by taking synonyms, lexicons and inflexions of these mappings.

Another database which is helpful in relation linking is Wordnet. It is a lexical database of semantic relations between words. Wordnet adds structure to relationships across the words. Wordnet has synsets so that almost or roughly synonymous words form a group in the same synsets. Every synset is connected to another synset by various relations used in the Wordnet. Nouns are connected by hypernyms(more abstract terms), hyponyms (more specific terms), meronym and holonym. Verbs present in the Wordnet are connected by hypernym, troponym, entailment and coordinate terms. Such structured information is beneficial for relation linking. There are tools such as ReMatch [102] which uses wordnet similarity for relation linking.

Another way to extend the vocabulary database for relation linking is extracting phrase-patterns from the text available on the internet. The information held by semantic triple is available in free text too, there have been several other works to extract phrase-patterns such that they have extensive coverage of the way a predicate is expressed in natural language. One such work has been done by [103] , by extending the BOA Framework. The goal of this work is to extract structured data as RDF from unstructured data. In achieving this goal, they published a BOA pattern library, where several phrase-patterns from the text are mapped to predicate labels.

A larger-scale relation alignment work has been done in the T-REx dataset [104], creating mappings between free-text documents and KG triples. TREx consists of 3.09 million Wikipedia abstracts aligned with 11 million Wikidata triples, covering more than 600 unique Wikidata predicates. T-REx is twice in order of magnitude larger than the most extensive available alignments datasets and covers 2.5 times more predicates.

3.4.3 Entity and Relation Linking in KGQA

Many QA systems use an out-of-the-box entity linker, often one of the ones as mentioned earlier. These tools are not tailor-made for questions and instead trained on large text corpora, typically devoid of questions. It may create several problems such as questions not spanning over more than one sentence, thereby rendering context-based disambiguation relatively ineffective. Further, graph-based systems rely on the presence of multiple entities in the source text and disambiguate them based on each other. This becomes difficult when dealing with questions, as they seldom consist of multiple entities.

Thus, to avoid the issues mentioned above, a variety of approaches have been employed for entity and relation linking for question answering. Semantic parsing-based systems such as TBSL [105] first link the entities and generate a list of candidate relations based on the identified resources. They use several strings and semantic similarity techniques to finally select the correct entity and relation candidates for the question. In these systems, the process of relation linking depends on linking the entities. Generating entity and relation candidates has also been explored by [78], which uses these candidates to create staged query graphs, and later re-ranks them based on the textual similarity between the query and the target question, computed by a Siamese architecture-based neural network. There are some QA systems such as Xser [106], which performs relation linking independent of entity linking. STAGG[78] takes the top 10 entities given by the entity linker and tries to build query-subgraph chains corresponding to the question. This approach considers a ranked list of entity candidates from the entity linker and then chooses the best candidate based on the query subgraph formed. Generally, semantic parsing based systems treat entity and relation linking as separate tasks which can be observed in the generalized pipeline of Frankenstein [6] and OKBQA `www.okbqa.org/`.

3.5 Natural Language Generation

NLG has been an active field of research since the last 70 years. Realization of NLG has been in different forms over the years, depending on the use-cases and resources available at the time. There are three basic techniques used in NLG realization: First, the templates based approach which generally works on filling in slots. These are suitable for small domains where less variation is required [McRoy et al., 2003].Second, the Hand-Coded Grammar-Based Systems, which offer an alternative choice to templates. They potentially can provide general-purpose, domain-independent NLG solutions. Third, the Statistical Approaches are recent, that have tried to obtain probabilistic grammars from large scale dataset using machine learning, reducing the manual labour required while increasing coverage.

In the scope of the thesis, we use NLG in the task of answer verbalization. Thus we cover related literature to this task and similar tasks. The three tasks we discuss in this section are SPARQL Query Verbalization, Triple Verbalization and Answer verbalization.

3.5.1 SPARQL Query Verbalization

To the best of our knowledge, there exists two query verbalization approaches exploring methods to translate SPARQL query to natural language (English): (1) SPARQL2NL [107, 108] and (2) SPARTIQULATION [109].

The pipeline of the **SPARTIQULATION** model is similar to the basic NLG architecture, as shown in Figure 3.5. The major difference lies in the approach of creating the document plan in the Document Structuring phase consisting of the following segments:

1. Query graph representation
2. Main entity identification
3. Query graph transformation
4. Message types
5. Document plan

In a nutshell, the textual SPARQL, SELECT queries are transformed into a graphical representation, called query graph. This generated query graph can perform traversal and transforms. The next step is to identify the primary entity, a variable, within the query graph, which portrays the subject of verbalization. After the identification of the primary entity, the query graph is transformed with the root of the primary entity. Now, this graph is split into messages which are independently verbalized. A set of message types

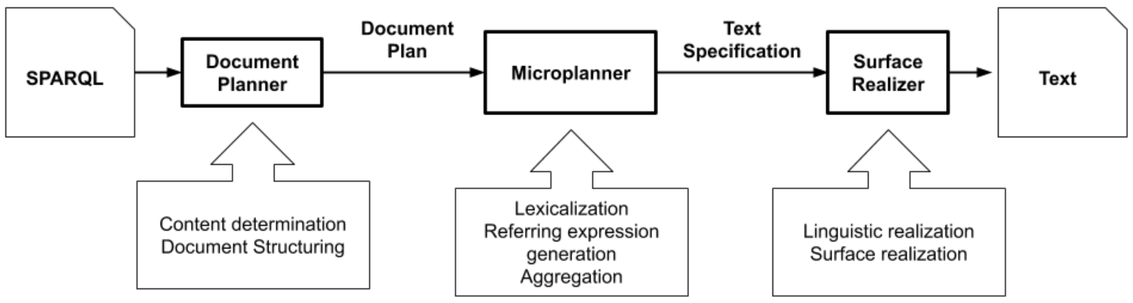


Figure 3.5: SPARTIQULATION Pipeline

were developed based on their role in the verbalization step, allowing a representation of the query graph. The final output of Document Planner, the first component of the pipeline, is the Document plan, which presents the ordering of messages according to their classes.

The basic steps in the **SPARQL2NL** approach consist of:

1. Pre-processing: The goal of this step is to transform any nested disjunctions and conjunctions present in the query into the disjunctive normal form (DNF) (*normalization*), and to collect information by processing the query and finding all graph patterns **?x rdf:type A** for every projection variable **?x** (*type extraction*).

2. Processing: In this step, the input query is divided into three parts viz. body, optional and modifier. Each of these parts is then assigned to a corresponding sentence tree. This yields a list of dependency trees for the query.

3. Post-processing: The idea behind this step is to transform the generated descriptions using a rule-based approach. The rules developed operate to cluster and order the input sentences.

4. Verbalization: The final step transforms the output of the previous steps into natural language.

3.5.2 RDF Triple Verbalization

The most common approach prevalent in QA system to generate answer in natural language is by verbalizing RDF triples. A list of resources to train such models are shown in Table 3.4.

Dataset	E2E [110]	WebNLG [111]	30M Factoid [11]
Domain	Restaurants	DBpedia	Freebase
Relations covered	8	373	1837
Size	50k+	21855	30M

Table 3.4: RDF triple Verbalization Datasets

E2E dataset [110] addresses information about restaurants containing over 50k dialogue-acts covering only eight relations. This dataset's limited domain knowledge renders it challenging to be implemented in open-domain QA systems.

WebNLG [111] and 30M Factoid [11] fetch information from knowledge graphs, such as DBpedia and Freebase respectively. The former dataset transforms RDF triple into natural language sentences, whereas the latter is used for generating natural language questions from RDF triples. The WebNLG challenge [112], based on the WebNLG dataset, provides an original benchmark for evaluation and assessing micro-planners. The next challenges focused on multilingualism to explore the morphological variations.

These datasets can be plugged into training a module of a QA system pipeline which converts the matching facts of a particular question into natural language. However, with the lack of knowledge of the question, the model would fail to learn key elements of the question for generating the answer. Thus, the generated answer contains information related to the fetched triples but not of the question. Consequently, it denies the user of the appropriate information needed to validate the answer.

3.5.3 Answer Verbalization

From the viewpoint of verbalization of answers, we characterize three resources, as shown in Table 3.5, currently in practice. To the best of our knowledge, these are the only few resources relevant for research in this domain.

Dataset	COREQA [113]	GENQA [114]	VQuAnD[5] [115]
Domain	In-domain	Open-domain	Open-domain
Knowledge Base	Restricted KB	Encyclopedia websites	DBpedia
Language	Chinese	Chinese	English
Baseline model	GENQA (Integer Linear Programming or ILP)	COREQA (COpying and REtrieving)	Standard Seq2Seq models

Table 3.5: Answer Verbalization Datasets

COREQA [113] and **GENQA** [114] are Chinese datasets resulting in scope for research for verbalization in English language. The former dataset provides QA pairs only in a restricted domain over merely 4 relations (*birthdate*: including *year*, *month* and *day*, and *gender*). While, the latter is a large-scale open-domain dataset collecting facts from Chinese community Encyclopedia websites[3] and extracting QA-pairs from two Chinese QA sites[4].

The recent dataset **VQuAnD**[5] [115] is the first and only dataset relevant for the task of verbalization in English. It contains questions extracted from LC-QuAD [**inproceedings2**] on DBpedia as the target KG. The advantage of this dataset is doubled due to the availability of SPARQL representation for questions. Thus, it is useful for model verbalizing answers to form a SPARQL query.

Different methodologies are used in these studies for handling the task in hand. Both **GENQA** and **COREQA** are not NLG models, rather are end-to-end QA pipeline comprising of a knowledge retrieval model and a generation model. The **GENQA** model, depicted in Figure 3.6, is based on Integer Linear Programming (ILP). It consists of *Interpreter*, *Enquirer*, *Answerer* and an *external Knowledge base*. In short, the *Interpreter* generates a formal query representation $\mathcal{H}_Q$ of the question Q and stores it in the *short-term memory*. Then, the *Enquirer* interacts with the *long-term memory*, in this case the *external KB*, to retrieve the relevant facts for the formal query $\mathcal{H}_Q$ and summarizes the results in the form of a vector r_Q. The *Answerer* is responsible for verbalizing the answer with the help of an *Attention Model*

[3]Baidu Baike, Baike.com, Douban.com
[4]Baidu Zhidao, Sogou Wenwen
[5]The dataset can be found: https://github.com/AskNowQA/VQUANDA

and a *Generator*. It takes in the formal query $\mathcal{H}_Q$, through the *Attention Model* as well as the summarized vector r_Q, to generate an answer A from the *Generator*.

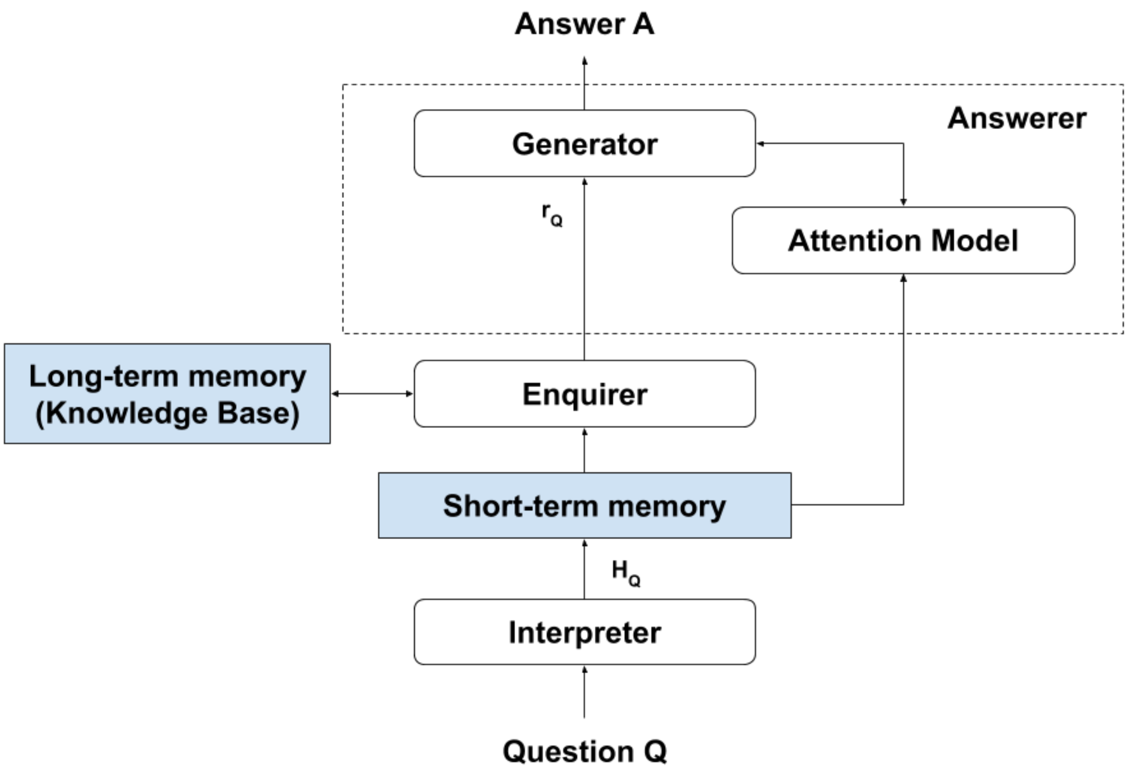


Figure 3.6: GENQA Workflow

The model architecture of **COREQA** utilizes the Encoder-Decoder framework, hooked up with a KB engineer. Firstly, the knowledge retrieval module is applied to fetch relevant triples or facts from the KB by analyzing the question. Then, the input question and the retrieved triples are encoded in the Encoder phase to generate a corresponding representation called the short-term memory. These encoded representations are sent to the Decoder, which generates the final answer in natural language. The decoder process poses these major distinctions from the attention-based Seq2Seq model [43]:

- *Answer Word Prediction*: **COREQA** predicts semantic units, such as words or tokens, of the answer in regard to a mixed probabilistic model with three modes: *predict-mode*, predicting words in the vocabulary, *copy-mode*, choosing tokens from the question, and *retrieve-mode*, choosing tokens from retrieved facts.

- *State-Update*: In this step, **COREQA** utilizes the predicted word at time-step $t-1$, its word embeddings and its positional attention information to update the next state s_t.

- *Reading short-term Memory*: The short-term memory representation of the question and matching facts are inputs to the **COREQA** in two formats: "meaning" with the help of embeddings and properties' values in terms of positional information.

The baseline models used for evaluation of the **VQuAnD**[6] dataset are conventional Seq2Seq models using attention mechanism [43, 116], CNN based Seq2Seq [117] and a transformer based neural architecture[7] [118]. However, the result of experiments of VQuAnD on these models show a lot of scope for improvement.

[6]The source code for the baselines can be found: https://github.com/endrikacupaj/VQUANDA-Baseline-Models

[7]For a comprehensive guide to transformers: http://nlp.seas.harvard.edu/2018/04/03/attention.html

CHAPTER 4

Intermediate Language based KGQA

With the advent of massive scale knowledge bases (such as DBpedia [51], YAGO [119], Freebase [53], Google Knowledge Vault [120], Microsoft Satori, etc.),the need to have a user-friendly interface for querying them becomes relevant. However, users usually are not deft in (and in most cases lack the knowledge of) writing formal queries. *Natural language query formalization* (*NLQF*) is a formal and systematic procedure of translating a user query in natural language (NL) into a query expression in a target formal query language. In this chapter, we scope the problem of NLQF to RDF/RDF-S knowledge bases only. Within this context, the target formal query language chosen is SPARQL [121] – the W3C recommended and widely adopted query language for RDF data stores.

NLQF into SPARQL for question-answering on RDF data stores is non-trivial. This can be attributed to several reasons:

1. a semantic interpretation of natural language query is intrinsically complex and error-prone,
2. the schema of the target dataset is not fixed,
3. a partial lack of rich schema structures of RDF datasets leading to syntactic mismatches,
4. lexical mismatches of query tokens, and
5. mismatches due to lack of explicit entailed relations in an RDF store.

One of the key linguistic challenges is the accurate identification of the *query desire* (also known as *query intent* or *answer type* in the literature) of a user-query. Another major challenge is that a query can be paraphrased into multiple forms, thereby triggering the potential of lexico-syntactic mismatches. Also, there is no unique way to create schemata in RDF, i.e. the same fact can be written in different triple forms and could also be expressed using multiple triples.

Research Question 1 (RQ1)

How does the semantic understanding of the user question affect the performance of KGQA?

Contributions of this chapter are summarize as follows:

- A novel paraphrase resilient query characterization structure (and algorithm), called *NQS*(Normalized Query Structure), is proposed. NQS is less sensitive to structural variation. It supports complex queries, hence, serves as a robust intermediary formal query representation.

- An NQS to SPARQL translation algorithm (and tool) is proposed that supports user queries to be agnostic of the target RDF store structure and vocabulary.
- An evaluation of AskNowNQS in terms of: (i) assessing robustness using the Microsoft Encarta data set, and (ii) evaluating accuracy using a community query dataset built on the OWL-S TC v.4.0 dataset and the QALD-4/5 datasets.

This chapter is based on the following publications ([15]):

1. **Mohnish Dubey**, Sourish Dasgupta, Ankit Sharma, Konrad Höffner, Jens Lehmann . "Asknow: A framework for natural language query formalization in sparql." In European Semantic Web Conference, pp. 300-316. Springer, Cham, 2016.

In this chapter, we propose a framework for natural language query formalization, called *AskNowNQS*, for posing queries in natural language (English in our case) to target knowledge graph. AskNowNQS uses an intermediate canonical syntactic structure, called *Normalized Query Structure* (*NQS*), into which an *NQS fitting* algorithm normalizes English queries.
The primary objectives of NQS are:

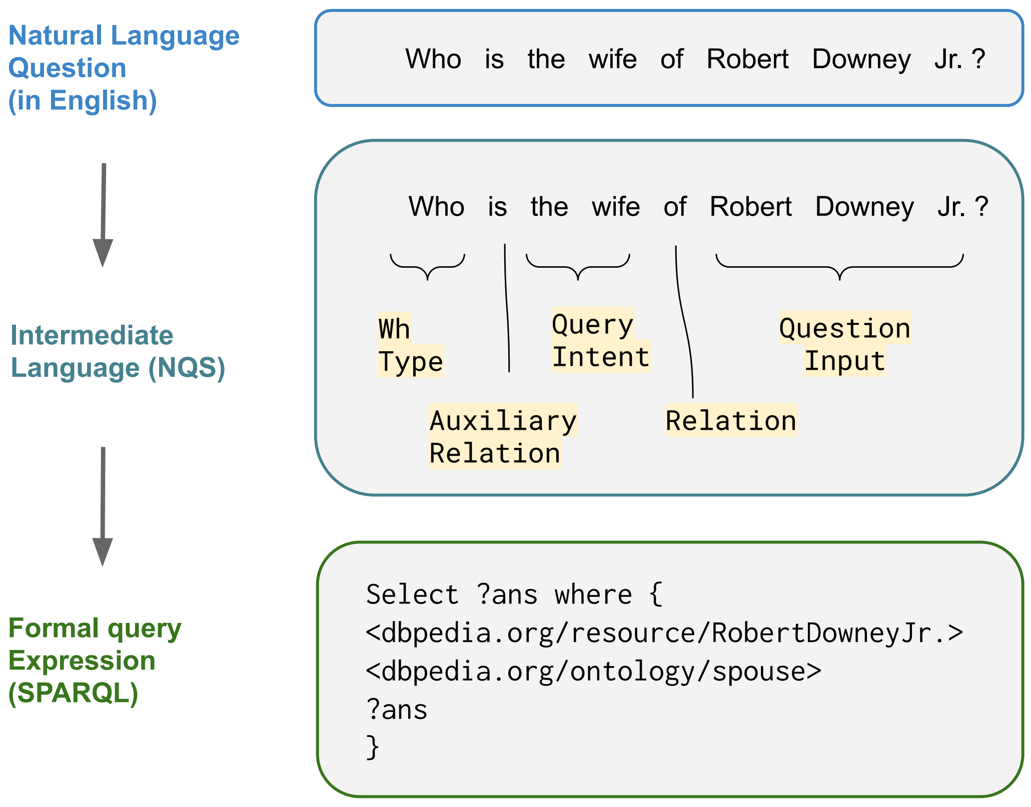
1. NQS works as an intermediate language between natural language and formal language. NQS captures the semantics of the user question by extracting question intents(desire) , inputs and relation between them.
2. Second objective of the NQS algorithm is to normalize paraphrased queries into a common structure.
3. Another objective is to help a SPARQL translator to easily identify the query desire, query input (i.e. additional information provided by the user), and their mutual semantic relation.

As an example, given the query: “*What is the capital of India?*", the algorithm will be able to differentiate the query desire (i.e. instances of class *capital*) and relate it to the input *India* via the relation *of*. Here, the input plays an important role in automatically constructing the declarative formal description of the desire. After a query is normalized into an NQS instance, the SPARQL translator then maps the query tokens in the NQS instance to entities defined in the RDF data store. This is done to solve the potential problem of lexical and schematic mismatch mentioned earlier. We show empirically, using QALD [122] benchmark datasets, that the devised NQS fitting algorithm is accurate in correctly characterizing (both in terms of syntax and semantics) most NL queries.
The chapter is organized in the following sections: (1)Preliminaries introduces to the definitions required for (2) Problem Statement, (3) *Approach*, where the formal notions of *NQS* query is defined, (4) in which the *Architectural Pipeline* of *AskNowNQS* is elaborated, (5) *Evaluation*, where various evaluation criteria are discussed.

4.1 Preliminaries

Definition 4.1.1 ***Question Input***:*Given a natural language question Q, question Input I is the information/subject provided by the user in the question. Input is the central subject around which question is framed by user.*

Example: for question "Who is the wife of Robert Downey Jr?", here the query Input *I* is ‘Robert Downey Jr‘, and the question has been framed around this subject.


Natural Language Question (in English)
Who is the wife of Robert Downey Jr. ?
Intermediate Language (NQS)
Who is the wife of Robert Downey Jr. ?
Wh Type
Query Intent
Question Input
Auxiliary Relation
Relation
Formal query Expression (SPARQL)

```
Select ?ans where {
<dbpedia.org/resource/RobertDowneyJr.>
<dbpedia.org/ontology/spouse>
?ans
}
```

Figure 4.1: Showcasing use of Intermediate Language in KGQA

Definition 4.1.2 ***Question Desire***:*Given a natural language question Q, Desire D is the user intent in the question. Desire is the expected answer by the user for the posed question.*

Example: for the question "Who is the wife of Robert Downey Jr?", here the user intends to know the wife of Robert Downey Jr, thus here wife is the intent.

Definition 4.1.3 ***Explicit Desire***: *Given a natural language question Q, explicit desire is identified directly in the question and requires no inference for hidden user intent.*

Example: for the question "What is the capital of USA", here the user intends to know the capital of United States of America, and the intent can be directly identified in the question phrase.

Definition 4.1.4 ***Implicit Desire***:*Given a natural language question Q, implicit desire is identified indirectly in the question and requires inference for hidden user intent.*

Example: for the question "What is Wikipedia?", here the user intends to know the definition or more details of Wikipedia, and the intent can not directly identified in the question phrase, requires inference at the surface level.

Definition 4.1.5 ***Simple query***: *A simple query consists of a single and unconstrained query-desire (explicit or implicit) and a single, unconstrained, and explicit query-input.*

For example, in “*What is the capital of USA?*" the query-desire (i.e. *Capital*) is explicit, single, and not constrained by any clausal phrase. The query-input (i.e. *USA*) is also explicit, single, and unconstrained. The query-desire can be implicit: For example, in the query “*What is a tomb?*", the implicit query-desire is the *definition* of *tomb*, while the query-input *Tomb* is single and unconstrained.

Definition 4.1.6 ***Complex query***: *A complex query consists of a single query-desire (explicit or implicit, constrained or unconstrained) and multiple, explicit query-inputs (constrained or unconstrained).*

For example, the query “*What was the capital of USA during World War II?*" is a complex query where the implicit, unconstrained query-desire is single (*Capital*) while the query-input is multiple (*USA*, *World War II*).

Definition 4.1.7 ***Compound query***: *A compound query consists of conjunction/ disjunction operator connectives between one or more simple or complex queries.*

An example of a compound query is “*What are the capitals of USA and Germany?*".

4.2 Problem Statement

4.2.1 Motivation

One of the critical challenges in question answering over knowledge graph is to tackle the wide gap between interpretation of natural language question and semantics of formal language query. To reduce this research gap, we propose to introduce an intermediate language between the natural language and formal expression. We take the motivation for the use of intermediate language from the use of intermediate language in different use-cases such as computer code compilation. The first purpose of the intermediate language is to provide a platform where the system could capture the user input and user intents from the natural language question. The second is to provide semantics to generate the formal query, such that it is a correct representation of the user question.

Our chosen and a popular formal query representation language is SPARQL, in which basic graph patterns consist of semantic triple comprising of subject, predicate and object. So, the primary objective of query parsing should be to identify the query desire/s and describe it in terms of the query predicate/s and query input/s. The identification of such a constraint relation is called *query desire-input dependency*. One of the key tasks for solving the problem of NLQF is to do *syntactic normalization* of NL queries. Syntactic normalization is the process which re-structures queries having different syntactic structural variations into a common structure so that subsequent formalization can be executed using a standard translation algorithm on this structure. Such normalization is difficult to achieve through a *query desire-input dependency* identification process alone. It is in this direction that we propose a chunker-styled *pseudo-grammar*, an intermediate language, called *Normalized Query Structure* (*NQS*).

4.2.2 Problem Formalization

A quick brief of the KGQA problem statement is that;
Given a language question Q_{NL} KGQA aims to model a transformation function f_{kgqa} such that :

$$f_{kgqa} : Q_{NL} \mapsto Q_{FL}$$

$$(Q_{NL})^{I_{NL}} \equiv (Q_{NL})^{I_{FL}}$$

I_{NL} is the linguistic Interpretation of the natural language question
$I_{FL}$is the semantic interpretation of the formal query.

Here in the scope of this chapter we establish, an intermediate state between Q_{NL} and Q_{FL} known as T. A T is a transitional state which holds linguistic understanding of the natural language question and also provide enough semantics to form a formal expression.

On the basis of the KGQA formalization, we could further formalize the Intermediate Language based KGQA approach in general.

Formalization: Intermediate Language based KGQA

Given a natural language question Q_{NL} Intermediate Language based KGQA aims to model a transformation function f_{kgqa} such that :

$$f_{kgqa} : Q_{NL} \mapsto T_{state} \mapsto Q_{FL}$$

$$(Q_{NL})^{I_{NL}} \equiv (T_{state})^{I_{NL},I_{FL}}$$

$$T_{state})^{I_{NL},I_{FL}} \equiv (Q_{NL})^{I_{FL}}$$

T_{state} is a intermediate transitional state holding understanding of Q_{NL} and semantic interpretation of the formal query.
I_{NL} is the linguistic Interpretation of the natural language question
$I_{FL}$is the semantic interpretation of the formal query.

4.3 Approach: Normalized Query Structure (NQS)

NQS is the proposed basis structure of natural language queries. It acts like a surface level syntactic template for queries, defining the universal linguistic dependencies (i.e. query desire-input dependency) between the various generic sub-structures (i.e. chunks) of any query. The primitive sub-structures of any query is the query token (which determines the query type), query desire, the query input, and the dependency relation that connects them. Each of the primitive sub-structures can be assigned a linguistic characterization (i.e. type). Example characterization are POS (part-of-speech) tag based chunks (such as noun phrase, verb phrase, etc.). For instance, desire and input can be hypothesized to assume a noun form, while the dependency relation can be assumed to be a verb form. An example of such characterization can be seen in the query: "*In which country is New York located?*". Here, the query desire is the noun phrase *country*, the query input is the noun phrase *New York*, and the dependency relation is the verb phrase *located in*. Since the number of query tokens is finite and there are only three query forms (simple, complex, compound), natural language queries can be categorized into a (finite) set of generic NQS templates.

We now introduce the NQS syntax definitions as follows:

Formalization: Simple Query NQS

A *simple query* can be characterized according to the following NQS structure:

$$[Wh]\ [R_1]\ [D]\ [R_2]\ [I]$$

here, $[D] = Q_D^? M_D^* D$ and $[I] = Q_I^? M_D^* I$
where the notation is defined as follows[1]:
$[D]$: Query desire class/instance-value is restricted to the following POS tags: *NN, NNP, JJ, RB, VBG*.
When and *where* queries have [D = NULL], and NQS automatically annotates D as *TIME* and *LOCATION* respectively.
$[I]$: Query input class/instance-value restricted to the POS tags: *NN, NNP, JJ, RB, VBG*.
$[R_1]$: Auxiliary relation - includes lexical variations of the set: {*is*, *is kind of*, *much*, *might be*, *does*}.
$[R_2]$: Relation that acts as (i) predicate having D as the subject and I as the object or (ii) action role having I as the actor - value restricted to the POS tags: *VB, PP, VB-PP*
$Q_D^?$ or $Q_I^?$: Quantifier of D or I - values restricted to the POS tag: DT. The ? indicates that Q can occur zero or one time before D or I.
M_D^* or M_I^*: Modifier of D or I - value restricted to the POS tags: *NN, JJ, RB, VBG*. The $*$indicates that M can occur zero or multiple time before D or I.

Characteristics of Relation Tokens: R_1 serves as a good indicator for resolving several linguistic ambiguities. For example, in a *how*-query, if R_1 is *much* (or its lexical variations) then it is a quantitative query. However, in a *who*-query if R_1 is *does* (or its lexical variations) then the associated verb is an activity (i.e. Gerund; ex: "*Who does the everyday singing in the church?*" - *everyday singing* is an activity in this case). R_2 is a relation that can either be associated with D as the subject or I as the subject but not both. If R_2 is positioned after D in the original NL query then R_2's subject is D. For example, in the simple NL query "*What is the capital of USA?*" the subject of R_2 (*of*) is D (*Capital*) and the object is I (*USA*). However, if R_2 is positioned after I in the original query then its subject is I. For example, in the query "*Which country is California located in?*" the subject of R_2 (*located in*) is I (*California*) and object is D (*Country*).

NQS of Complex & Compound Wh-queries: A complex *Wh*-query can be characterized according to:

$[Wh]\ [R]\ [D]\ [Cl_D^?]\ [R_2]\ [I_1^1]\ [([CC]\ [I_2^1])^*]^*...\ [Cl_2^?]\ [R_3]\ [I_1^2]\ [([CC]\ [I_2^2])]^*...$
$...[Cl_N^?]]\ [R_{N+1}]\ [I_1^N]\ [([CC]\ [I_2^N])]^*$
where:
Cl_D : clausal lexeme (constraining D). Example of clausal lexemes: *wh*-tokens, *that*, *as*, *during /while /before /after*, etc. It is to be noted that clausal lexemes generates nested sub-queries which themselves may (or may not) be processed independently to the parent query. An example where the sub-query (in bold) has a dependency is: "*Which artists where born on the same date as* ***Rachel Stevens****?*"
Cl_2: second clausal lexeme (constraining I_1)
Cl_k: clausal lexeme associated with k-th sub-structure
$[CC]$: conjunctive/disjunctive lexeme for I
$[D]$: query desire - value restricted to POS tags: {*NN, NNP, JJ, RB, VBG*}
$[I_l^k]$: l-th query input for k-th structure - value restricted to POS tags: {NN, NNP, JJ, RB, VBG}

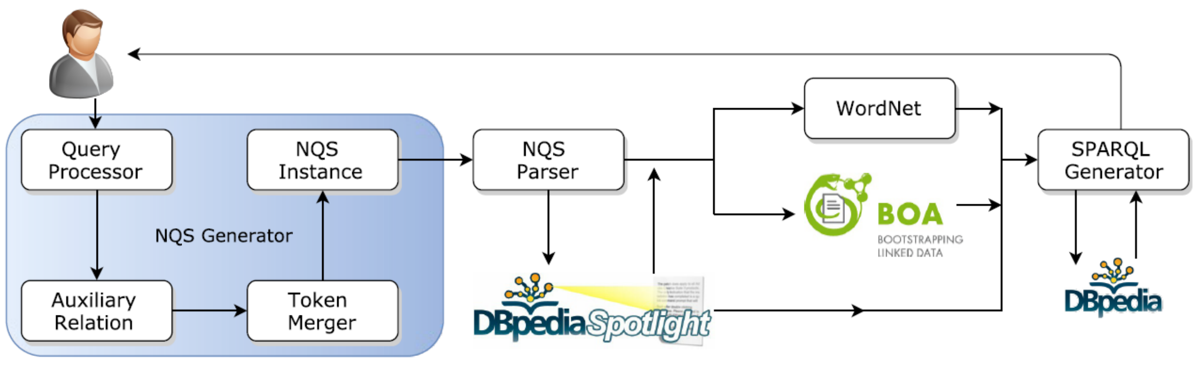


Figure 4.2: Architectural Pipeline of AskNowNQS

$[R_{k+1}]$: relation associated with the k-th clause that acts as (i) predicate of D as the subject and I as the object or (ii) action role of I as the actor - value restricted to to POS tags: {VB, PP, VB-PP}.
Notation with ? may occur zero or one time.
Notation with $*$ may occur zero or multiple time. $[M]$: modifier of the D or the I - value restricted to POS tags: {NN, JJ, RB, VBG}. The * indicates that M can recur before D or I.

In the given complex NQS, we see the possible repetition of the structure: $[I_1^k][([CC]$ $[I_2^k])]$. Within this structure, there is an optional substructure $[([CC][I_2^k])]$ that may add to the number of inputs within each of such structures. A clausal lexeme in a complex clausal *wh*-query is always associated with such a structure. The number of clausal lexemes is the same as the number of such structures in a given query. It should be noted that there must be at least two such structures for a query to qualify as complex. Clausal lexemes are optional and hence, the NQS also works for complex non-clausal wh-queries. We name the following structure as *clausal structure* (*CS*):
$[Cl_D^?]\ [R_2^?]\ [I_1^1]\ [([CC][I_2^1])^*]^?...[Cl_2^?]\ [R_3^?]\ [I_1^2]\ ([CC]\ [I_2^2])^?...$
$...([Cl_N^?][R_{N+1}^?][I_1^N]([CC][I_2^N])^*[?].$

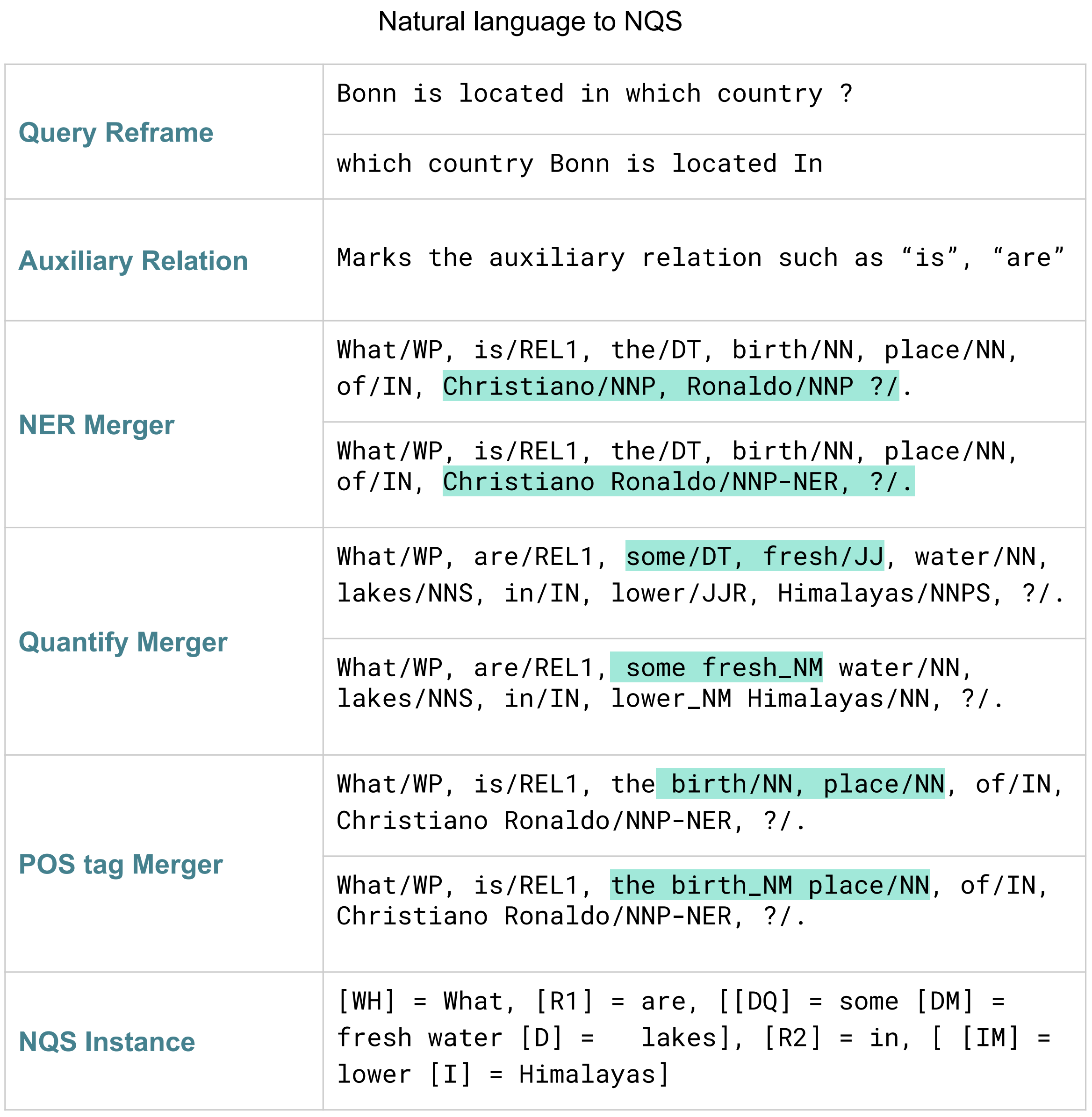
A compound *Wh*-query can then be characterized according to:
$[Wh^1]\ [R_1^{1?}]\ [D_1^{1?}]\ [Cl_D^?]\ [R_2^?]\ [I_1^1]\ ([CC]\ [I_2^1])^*\ [Cl_2^?][R_3^?][I_1^2]([CC][I_2^2])^*....$
$([Cl_N^?][R_{N+1}^?][I_1^N]([CC][I_2^N])^*)^*[?].$

4.4 AskNowNQS Architectural Pipeline

We have outlined the architectural pipeline of AskNowNQS, in Fig. 4.2 with two basic components: the NQS Instance Generator and the NQS to SPARQL converter.

4.4.1 NQS Instance Generation

As mentioned in the previous section, the objective of NQS is not to propose yet another grammar but rather to provide a modular format to the internal sub-structures of a query. Therefore, an efficient template-fitting algorithm that can parse the natural language query, identify the sub-structures (using a standard POS tagger), and then fit them into their corresponding *cells* within the larger generic NQS template is required. Our proposed template-fitting algorithm is called *NQS Instance Generator*. Through the fitting process the query-desire, query-input, and other relevant information can be extracted. A fitted NQS is called an *NQS instance*. The fitting process automatically leads to normalization. Also, it is



Natural language to NQS

Query Reframe	Bonn is located in which country ?
	which country Bonn is located In
Auxiliary Relation	Marks the auxiliary relation such as “is”, “are”
NER Merger	What/WP, is/REL1, the/DT, birth/NN, place/NN, of/IN, Christiano/NNP, Ronaldo/NNP ?/.
	What/WP, is/REL1, the/DT, birth/NN, place/NN, of/IN, Christiano Ronaldo/NNP-NER, ?/.
Quantify Merger	What/WP, are/REL1, some/DT, fresh/JJ, water/NN, lakes/NNS, in/IN, lower/JJR, Himalayas/NNPS, ?/.
	What/WP, are/REL1, some fresh_NM water/NN, lakes/NNS, in/IN, lower_NM Himalayas/NN, ?/.
POS tag Merger	What/WP, is/REL1, the birth/NN, place/NN, of/IN, Christiano Ronaldo/NNP-NER, ?/.
	What/WP, is/REL1, the birth_NM place/NN, of/IN, Christiano Ronaldo/NNP-NER, ?/.
NQS Instance	[WH] = What, [R1] = are, [[DQ] = some [DM] = fresh water [D] = lakes], [R2] = in, [[IM] = lower [I] = Himalayas]

Figure 4.3: Step-wise breakdown of the Translation from Natural Language Question to NQS

resilient to paraphrasing of queries since the sub-structures in a paraphrase typically remain unaffected[2]. The change is only in the inter sub-structure positioning (eg: “*New York is located in which country?*" vs. ‘‘*Which is the country where New York is located?*")[3]. Note that the original NL query may lose its syntactic structure during the NQS instance generation process and also, it does not guarantee the grammatical correctness of the normalized query.

[2]In certain cases minor splitting has to be handled in the dependency relation, where the query starts with a preposition, e.g.: “***In** which country is New York **located**?*"

[3]Paraphrasing may include lexical substitution of synonymous query tokens and morphological changes of the tokens.

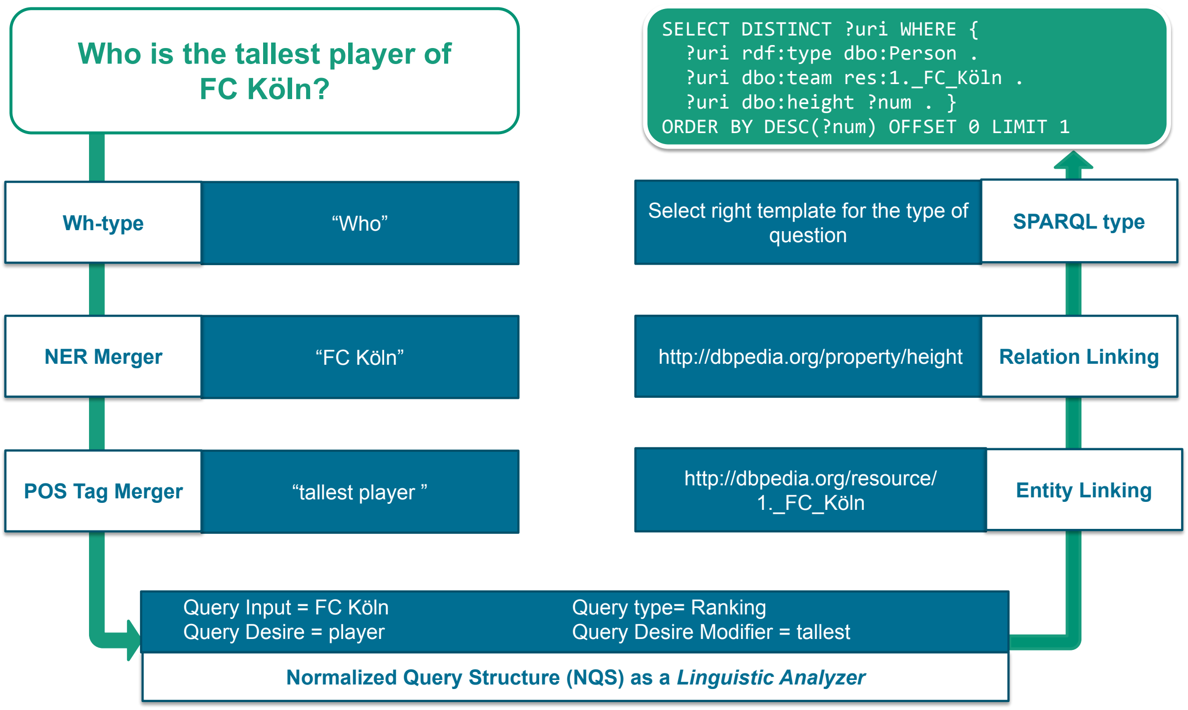
In summary, the flexibility of NQS modeling is to be attributed to the NQS Instance Generator algorithm. All internal components of the NQS Instance Generator are described as follows:

Query Processor: This module initiates the NQS query processing system by initializing other modules. It calls the POS-tagger (in our case we used Stanford coreNLP[4]) so as to tag every query token. Then it breaks the query text into individual POS-tagged query tokens. Subsequently, the *Syntactic Normalizer* transforms original queries to have a common syntactic structure. For example, it normalizes each query to start with *wh*-token, handling apostrophe, etc.
Auxiliary Relation Handler: The module to extract $R1$. More details regarding the utility of the auxiliary relation is given in previous section.
Token Merger: This module merges (or *chunks*) tokens that together form a single meaningful lexeme. Based on the POS tags of the tokens in the original NL query, the token merging module can guess the possible tokens to be combined so that they can fit the NQS. For example, when the query, "*Who is the Prime Minister of India?*" is passed to the POS-Tagger we get the resulting answer: "Who_{WP} is_{VBZ} the_{DT} $Prime_{NNP}$
$Minister_{NNP}$ of_{IN} $India_{NNP}$?". Then two *consecutive tokens* are taken at a time and checked, using a token-merging map, whether they can be combined or not. We have manually bootstrapped the token-merging map on different types of token-pair lexico-syntactic patterns, using the M.S. Encarta 98 query dataset. The map keeps getting updated as and when other valid token-pairs are identified in future. Merging avoids unnecessary breaking of lexemes during the template fitting process, thereby improving efficiency. However, merging is a challenging task as it cannot be context-free. It is quite possible that two tokens may have to be merged in one query while it should not in another.
NQS Instance Generator: After the individual chunks have been identified, the query then goes through the NQS Instance Generator (see Fig. 4.2). It uses the following two hypothesis about the generic structure of a query (which was observed to be empirically true when tested on Microsoft Encarta 98, which is a large-scale query dataset).

Hypothesis 1: The query desire is always a noun phrase.
Hypothesis 2: The query desire always precedes the query input in the normalized NL query.

The algorithm also utilizes the characteristics of desire-input dependency relations, as discussed in the previous section. Every time it encounters a noun phrase chunk it treats it as a candidate desire. Depending upon the availability of conjunctive connectives, it then does a conflict resolution among all candidate desires by verifying the positioning of the verb phrase. As an example, the query "*Desserts from which country contain fish?*" has three candidate desires: *dessert*, *country*, *fish* (based on Hypothesis 1). The main relation *contain* is positioned after *country*. Therefore, the *potential* subject of *contain* is identified to be *country*[5]. Now according to Hypothesis 2, the desire must precede the input in the NQS instance. So *fish* is resolved not to be a candidate desire any more, but rather an input. Now, the query has another main relation *from*, the subject being *dessert* and the object being a query token *which*. Thus, the algorithm resolves that *country* is the desire while *dessert* is another input. Finally, the algorithm analyzes that *country* being the desire, and also having the inverse relation *from* to the input *dessert*, cannot have the relation *contain* to the second input *fish*. Therefore, it is the input *dessert* which is the true subject of the relation *contain* to the object (i.e. the second input) *fish*. The final NQS will be: $[wh = which][R_1 = null][D = country][R_2 = from][I_1 = dessert][R_3 = contain][I_2 = fish]$. It is to be noted that there is an implicit nested dependent sub-query: "*Which desserts contain fish?*" because of the clausal connective *whose* (*Which country* ***whose*** *desserts ...*) that is an inverse of the relation *from*. This

[4]http://nlp.stanford.edu/software/tagger.shtml
[5]A standard dependency parser could also be used to understand the subject of the relation *contain*.


Who is the tallest player of FC Köln?
Wh-type
"Who"
NER Merger
"FC Köln"
POS Tag Merger
"tallest player "
Query Input = FC Köln
Query Desire = player
Query type= Ranking
Query Desire Modifier = tallest
Normalized Query Structure (NQS) as a *Linguistic Analyzer*
1._FC_Köln
Entity Linking
http://dbpedia.org/resource/1._FC_Köln
Relation Linking
http://dbpedia.org/property/height
SPARQL type
Select right template for the type of question

```
SELECT DISTINCT ?uri WHERE {
  ?uri rdf:type dbo:Person .
  ?uri dbo:team res:1._FC_Köln .
  ?uri dbo:height ?num . }
ORDER BY DESC(?num) OFFSET 0 LIMIT 1
```

Figure 4.4: Example Illustration of the AskNowNQS Pipeline

example illustrates that the previously outlined NQS syntax definitions are not static templates, but rather dynamically fitted.

4.4.2 NQS to SPARQL Conversion

Given an NQS instance, the NQS2SPARQL module translates it to a SPARQL query and returns the result from the SPARQL endpoint. There are four main steps in this module as shown in Algorithm 1.

NQS analysis

Once we have an NQS instance for a query, the system treats it as per its category. The categories are the expected query types, specifically: i) Boolean ii) Ranking iii) Count iv) Set (List) and v) Property Value. In a Boolean query a user asks whether a specific statement is True of False. For instance "Is Barack Obama a democrat?" A Ranking query requires ranking the answers based on some entity dimensions, e.g. "Which is the highest mountain in Asia?". In a Count query the user intent is to get the number of times a certain condition is repeated. A Set query will generate a list of items which satisfy a required condition. In a Property value query the user intent is to ask for the value of a property of the given input. As an example, in the query "What is the capital of India?" the user intents to extract the value of the property "capital" given the input "India". Query-types are chosen based on desire (D) and wh-type (wh) of the NQS instance. Each category is processed by a different SPARQL query syntax converter.

Algorithm 1 NQS to SPARQL Algorithm.

```
input  : NQS instance $\aleph$, knowledge base $KB$
output : SPARQL query results
// Step 1: NQS analysis
$D \longleftarrow$ queryDesire($\aleph$)
    $wh \longleftarrow$ getWhQuestionType($\aleph$)
    $t \longleftarrow$ determineQueryType($D$, $wh$)
    $I \longleftarrow$ queryInput($\aleph$)
    // Step 2: SPARQL preparation
$i \longleftarrow$ mapInput($I$)  $S = \{(p, v)|(i, p, v) \in KB\}$;        // construct predicate object map
init $p_{match}$  foreach $(p, v) \in S$ do
    // label matching
    if $lm(p)$==$D$ then $p_{match} = p$; break;
    // WordNet synonyms of desire
    if $wns(D)$ ==$p$ then $p_{match} = p$; break;
    // BOA library
    if $BOA(D)$ ==$p$ then $p_{match} = p$; break;
end
// Step 3: SPARQL generation and retrieval
$q \longleftarrow$ generateQuery($i$, $p_{match}$, $KB$)
 $R \longleftarrow$ executeQuery($q$, $KB$)  return $R$
```

Entity Mapping

The basic operation here is to retrieve the knowledge base entity matching the spotted query desire, query input and their relation. For the QALD experiments described later, we annotated the query using DBpedia Spotlight [123]. As a result of the mapping, we get the knowledge base entity equivalent of the query input I which has been identified in the NQS instance. We denote this entity as i. The mapping approach then collects properties related to i (where i is a resource) and their values in set (denoted S).

Subsequently, each element (a pair of property and value) of S is observed. The next goal is to identify the entity which matches the desire (itself denoted as D) and denote it as d. This is done using three mapping functions as follows: The first test is made by a simple label matching function(lm). If this fails, then the second test for mapping($D \rightarrow d$) is through the WordNet synonym (wns)function. It finds the synonym of user desire using WordNet [124] within set S. If this test fails we move to next test. Here we use BOA pattern library [125] for the same purpose. When this is unsuccessful, then we declare that the query is unprocurable by the system.

SPARQL Generation

This component creates the final SPARQL query using information provided by above two steps. NQS analysis basically gives the SPARQL pattern possible. Where as i, d provide the key DBpedia information (vocabulary) required for SPARQL. Examples are given in Table 4.1. Currently NQS2SPARQL is functional for DBpedia only. However, we can plugin any other RDF store using suitable corresponding entity mapping module.

NQS to SPARQL

NQS Parser	`Decide query type : Boolean, Count, Ranking, List, Data property`

DBpediaSpotlight	`Query Input (I) maps to DBpedia equivalent`
	`[I]= Himalayas` $\longleftrightarrow$ `dbpedia.org/resource/Himalayas`
	`Dbpedia Spotlight Annotation`

WordNet	`To get synonym of Relation (between` *`Query Input`* `and` *`Desire`*`)`
	`[D]=writer` $\rightarrow$ `author` $\rightarrow$`dbpedia.org/ontology/author`
	`WordNet Synonyms`

BOA Pattern Library	`[[D]=wife]` $\longleftrightarrow$ `http://dbpedia.org/ontology/spouse`
	`Semantically similar patterns from web`

POS tag Merger	`Final SPARQL query based on above steps is generated`

Figure 4.5: Step-wise breakdown of the Translation from NQS to SPARQL

4.5 Evaluation

4.5.1 Evaluation Goal and Metric

Goal I. Syntactic Robustness:

Syntactic robustness of *NQS* measures its structuring capacity after normalization. Ideally, the *NQS* algorithm should be correct By *correct structuring* we mean that there should not be any mismatch between the POS tag of a linguistic constituent and its corresponding *NQS* cell. At the same time, the algorithm should be complete (i.e. there should not be any valid English query that is not accepted by the algorithm, either fully or partially). To evaluate robustness we decided on a simple measure called *Structuring Coverage* (*SC*). We measure SC in the following three different perspectives:
(i) SC-Precision: Given a test set of NL queries, *SC-Precision* is calculated as the ratio of the number of correct *NQS-structured* queries (N_{CI}) and the total number of *NQS-structured* queries in the test set (N_I). It largely depends upon the accuracy of the POS tagger used.
(ii) SC-Recall: Given a test set of NL queries, *SC-Recall* is calculated as the ratio of the number of correct *NQS-structured* queries (N_{CI}) and the total number of queries in the test set (N).

NL Query	List down all the Swedish holidays
NQS values	[WH = What], [R1 = is], [D = list], [R2 = of], [M = Swedish], [I = holiday]
Type	List
SPARQL	SELECT DISTINCT ?uri WHERE { ?uri rdf:type dbo:Holiday. ?uri dbo:country res:Sweden }
NL Query	In which country is Mecca located?
NQS values	[WH = which], [R1 = is], [D = country], [R2 = located In], [I = Mecca]
Type	Property Value
SPARQL	SELECT ?num WHERE { res:Mecca dbo:country ?num . }
NL Query	How many ethnic groups live in Slovenia
NQS values	[WH = How many], [R1 = *null*], [D = count(ethnic group)], [R2 = live in], [I = Slovenia]
Type	Count
SPARQL	SELECT COUNT (DISTINCT ?uri) WHERE { res:Slovenia dbo:ethnicGroup ?uri . }
NL Query	Who is the heaviest player of the Chicago Bulls?
NQS values	[WH = Who], [R1 = is], [M = heaviest], [D = player], [R2 = of], [I = the Chicago Bulls]
Type	Ranking
SPARQL	SELECT DISTINCT ?uri WHERE { ?uri rdf:type dbo:Person . ?uri dbo:weight ?num . ?uri dbp:team res:ChicagoBulls} ORDER BY DESC (?num) OFFSET 0 LIMIT 1

Table 4.1: QALD-5 Example on AskNowNQS

(iii) SC-F1: The Simple Harmonic Mean of *SC-Precision* and *SC-Recall*.

Goal II. Sensitivity to Structural Variation:

Sensitivity to structural variation of *NQS* measures the degree to which *NQS* can *correctly fit* queries having same *desire* (and its relationship with *input*) yet different syntactic structures. To evaluate *sensitivity to structural variation* we introduce following two measures:
(i) Variational-Precision (*VP*): Given a test set of NL queries, the *VP* is calculated as the ratio of the number of correct *NQS-structured* queries (i.e. without any of their variations getting incorrectly fitted) (N_{VI}) and the total number of identified queries in the test set (N_I).
(i) Variational-Recall (*VR*): Given a test set of NL queries, the *VR* is calculated as the ratio of N_{VI} and total number of queries in the test set (N).

Goal III. Semantic Accuracy:

Semantic accuracy of *NQS* measures the degree to which the query *desire* and its relation with query *inputs* has been properly identified. To evaluate this we use the following measures:
(i) Semantic-Precision (*SP*): Given a test set of NL queries, the *SP* is calculated as the ratio of the number of correctly identified queries (i.e. in terms of *desire-identification*, *input-identification*, and *desire-input relation identification*) (N_{SI}) with respect to a human-judgment benchmark, and the total number of identified queries in the test dataset (N_I).
(ii) Semantic-Recall (*SR*): Given a test set of NL queries, the *SR* is calculated as the ratio of N_{SI} with respect to a human-judgment benchmark, and the total number of queries in the test dataset (N).
Here are examples to give a better understanding of purpose of each measure:
Failed NQS (i.e. no instance):[Wh= NULL] [R1= is] [D= Berlin] [R2= NULL] [I= country][?]
Incorrectly structured NQS instance: [Wh= In which country] [R1= is] [D= Berlin] [R2= located] [I= NULL]. This will be considered as identified query (i.e. one in N_I).
Correctly structured NQS instance (i.e. in $N_C I$): [Wh = Which] [R1 = is] [D = Berlin] [R2 = located in] [I = country]. We use SC (and also VP, VR) to test $N_C I$ with respect to N_I and total queries (N).
Correctly "identified" NQS instance (i.e. in $N_S I$): [Wh = Which] [R1 = is] [D = country] [R2 = located in] [I = Berlin][?]. We use SP and SR to test this.

Goal IV Accuracy of the AskNowNQS System:

The final goal of the evaluation is to test the system on the QALD-5 [72] benchmark (Multilingual question answering over DBpedia). Here, we have queries in English language which are answered with NQS translated SPARQL.

4.5.2 Datasets

In order to evaluate *syntactic robustness* (for goal-I), we have used the Microsoft Encarta 98 [6] query test set. . The test set contains 1365 usable English *wh*-queries. There are total 522 queries of procedural *how* and *why* that have been excluded. We also created an extensive query set based on OWLS-TC v4[7] for evaluation of both *sensitivity to structural variation* (goal-II) and *semantic accuracy* (goal-III). Three research assistants independently formulated wh-queries for every web service of OWLS-TC v4 dataset, such that the query desire matches the given service output, and the query input matches the required service input. We had 1083 services to make three different query versions for each service. Similar syntactic structure queries were excluded resulting in a total of 2217 queries It is to be noted that the goal of the experiment (cf.: Goal II) was to test the robustness of an NQS Instance Generator, in terms of POS-tag pattern fitting (i.e. syntactic accuracy), over different syntactic variations of the same query. The OWLS-TC dataset consists of service descriptions of 1083 web-services from 9 different domains. A service description is a formal specification of the behavior of a web service in terms of its required input parameters, given output parameters, and other binding parametric details for runtime execution. The description also contains a short NL narrative of the overall behavior. Three different *wh*-queries are formulated for every service such that the query desire matches the given output of the service and the query input matches its required input. 90% of the queries were complex or compound queries. Ideally,

[6]http://research.microsoft.com/en-us/downloads/88c0021c-328a-4148-a158-a42d7331c6cf/
[7]http://projects.semwebcentral.org/projects/owls-tc/

	QALD 5			**M.S. Encarta**			**OWL-S TC**			**Total**		
	N	N_I	N_{CI}	N	N_I	N_{CI}	N	N_I	N_{CI}	N	N_I	N_{CI}
How	31	31	31	165	158	158	4	4	2	200	193	191
What	37	37	37	406	392	392	1711	1709	1608	2154	2138	2037
When	12	12	12	39	35	35	0	0	0	51	47	47
Where	5	5	5	85	82	82	20	20	19	110	107	106
Which	81	81	81	5	5	5	316	316	308	402	402	394
Who	48	48	48	143	143	143	166	166	166	357	357	357
Total	214	214	214	843	815	815	2217	2215	2215	3274	3244	3226

Table 4.2: *SC* Evaluation on Different Datasets

the extracted query desire by NQS should be semantically equivalent to the output parameter of the corresponding web service specification. Based on this notion, we have calculated *SC-accuracy*, *VP/VR*, and *SP/SR* for each of the three versions of query dataset. We also used the QALD-5 [72] datasets for Goal-IV and QALD-4 [66] for evaluating Goal-II. QALD-4 has released set of 24 query over three data sets: (i) *Drugbank*, (ii) *Diseasome*, (iii) and *Sider*. We created three versions of each of these 24 queries, that were syntactic variations having equivalent semantics.

4.5.3 Results

Result I. Syntactic Robustness:

We first performed the evaluation of *structural robustness* in terms of SC-Accuracy over different *query*-types on *Microsoft Encarta 98* dataset. We observe 100 % SC-Precision for all types of wh-queries, which shows that the NQS is theoretically sound. The *SC-Recall* came out to be 96.68 %. We then performed the same experiment over different *wh*-types on 2 more datasets: Training set of QALD-5's Multilingual tract (only *english* queries) and *OWLS-TC*. We observed a high overall SC-F1 of 98.99 %. The evaluation results are given in Table 4.2 and 4.3.

Result II. Sensitivity to Structural Variation:

We performed evaluation of *sensitivity to structural variation* of *NQS* over the OWL-S TC query dataset (three versions) and the QALD-4 dataset (three versions). *NQS* was able to correctly fit 919 out of the 1083 OWLS-TC queries (along with all their syntactic variation), giving high *VP* of 96.43 %. All 24 out of 24 QALD-4 queries, with all there syntactic variations, were correctly fitted in *NQS*, giving a high sensitivity to structural variation.

Result III. Semantic Accuracy:
We observed (refer Table 4.4) an *SP* of 91.92 % for the OWL-S TC query dataset. For QALD-4 dataset, it was observed that 21 out of 24 queries (with their variations) were correctly fitted in *NQS*. Analysis of the fail case clearly indicates that NQS failure is dependent upon syntactic and POS Tag failures.

	Result		
	SC_R	SC_P	SC_{F1}
How	95.50	98.96	97.20
What	94.57	95.28	94.92
When	92.16	100	95.92
Where	96.36	99.07	97.70
Which	98.01	98.01	98.01
Who	100	100	100
Total	98.53	99.45	98.99

Table 4.3: *SC* Evaluation on Different Datasets

Dataset	N_{Wh}	N_I	N_{VI}	N_{SI}	***VR%***	***VP%***	***SR%***	***SP%***
OWL S TC	1 083	953	919	876	84.85	96.43	80.88	91.92
QALD-4	24	24	24	21	100	100	87.50	87.50
Total	1 107	977	943	897	85.18	96.51	81.03	91.81

Table 4.4: Evaluation of Sensitivity to Structural Variation and Semantic Accuracy

	Processed	Right	Partial	Recall	Precision	F_1	F_1 Global
Xser	42	26	7	0.72	0.74	0.73	0.63
AskNowNQS	27	16	1	0.63	0.60	0.61	0.33
QAnswer	37	9	4	0.35	0.46	0.40	0.30
APEQ	26	8	5	0.48	0.40	0.44	0.23
SemGraphQA	31	7	3	0.32	0.31	0.31	0.20
YodaQA	33	8	2	0.25	0.28	0.26	0.18

Table 4.5: Results on the QALD 5 Benchmark.

Results IV. Accuracy of AskNowNQS:

We used the benchmark data set of the 5th Workshop on Question Answering over Linked Data (QALD), which defines 50 questions to DBpedia and their answers. Here we compare the our results with the result published by QALD-5 [72] (refer Table 4.5). Out of 50 questions provided by the benchmark we have successfully answered 16 correct and 1 partially correct. There were 5 questions where NQS algorithm fails to correctly identify the Inputs and Desire hence they could not be answered by translating them into SPARQL. The failure analysis of Result IV are as follows:
NQS failure: Queries where NQS failed were not further processed successfully. NQS failed only 5 times, which was due to incorrect dependency analysis.
Entity Mapping: There are 13 questions where AskNowNQS could not map the DBpedia equivalent of correctly identified input and desire. In some cases, the correct mapping was presented but insufficient to answer the query. As an example, the query *"Who killed John Lennon?"* is correctly processed by NQS and forwarded to DBpedia Spotlight for annotation. It maps *JohnLennon* to `http://dbpedia.org/resource/John_Lennon` which is a correct mapping in general terms. But we can not answer the question based on this resource. For that we would require `http://dbpedia.org/resource/`

Death_of_John_Lennon.
Relation Mapping: In some cases, system could not resolve the $R2$ (relation between input and desire) to the correct DBpedia property. Relations such as *study* and *graduated* were not mapped to the required DBpedia property *almaMater*.

4.6 Conclusion

In this chapter, we propose *AskNowNQS*, a KGQA framework, based on an intermediate language and novel syntactic structure *Normalized Query Structure* (NQS). NQS identifies the questions' inputs, intents (desire) and the relationship between them. The goal of the NQS is to assist the process of formal query construction. We empirically show, using benchmark datasets, that NQS is robust in terms of syntactic variation, and also highly accurate in identifying the query intents (along with its relationship to the query input). Hence, we show that NQS serves as a robust intermediary model for translating NL queries into formal queries. We have empirically demonstrated this by converting NQS to SPARQL. NQS in the current implementation does not support relations which are implicit in the question. In our performance analysis, it is evident that the major loss in accuracy is due to incorrect entity linking and relation linking. This analysis also shows that the entity linking system such as DBpediaSpotlight does not have their optimum performance on questions as they are short strings compared to long paragraph text. Thus, it highlights the research gap of not having NLP components designed explicitly for KGQA.

CHAPTER 5

Generating a Large Scale Dataset for KGQA

The previous Chapter presented a solution to KGQA by introducing an intermediate language between the natural language and formal language. This approach is rule-based on linguistic heuristic and involves expert domain knowledge. However, using machine learning approaches on such a task would certainly be the move forward, as we have witnessed that generally, machine learning approaches surpass the rule-based systems. The major hurdle for using machine learning approaches in KGQA is scarcity of large scale dataset with complex questions with formal expressions. There are large size datasets for simple questions but not on complex questions. This Chapter is dedicated to setting up a framework for generating large scale dataset of complex question on a knowledge graph with formal expression.

In other research fields, large scale data sets resulted in improvements in accuracy and precision as more modern machine learning and deep learning solutions could be applied to a research problem. Computer vision made progress by leaps and bounds in various tasks, as large size datasets for those tasks were published. ImageNet is a large scale dataset for the computer vision task of visual object recognition and it has more than 15 million annotated images for the task. In 2012 convolutional neural network (CNN) called AlexNet won the task, however, later it was outperformed by Microsoft's very deep CNN model. This establishes the fact that a large scale dataset helps the research community to grow and advance the state of the art. Generating large-scale dataset for knowledge graph question answering is a difficult task. One can only formalize a natural language question after having field expertise and having skill-set of knowledge graph understanding, formal query writing and the semantic web. KGQA's natural language questions are also limited to information captured in the KG already; thus, the questions need to be in the scope of the KG.

Lack of large size complex KGQA dataset is a challenge in the KGQA research area. In this chapter we will focus on how to overcome this challenge.

Research Question 2 (RQ2)

How to generate a large size Complex question dataset for KGQA, with a wide variety of questions, so the Machine Learning could leverage over the dataset.

Contributions of this chapter are summarize as follows:

1. A framework named LC-QuAD (Large-Scale Complex Question Answering Dataset), for generating NLQs and their SPARQL queries which reduces the need for manual intervention.

2. A dataset of 5000 questions with their intended SPARQL queries for DBpedia. The questions exhibit large syntactic and structural variations.

This Chapter is based on the following publications([16, 126])

1. Priyansh Trivedi, Gaurav Maheshwari, **Mohnish Dubey**, and Jens Lehmann. "LC-QuAD: A corpus for complex question answering over knowledge graphs." In International Semantic Web Conference, pp. 210-218. Springer, Cham, 2017. The first three authors contributed equally.

2. Gaurav Maheshwari, **Mohnish Dubey**, Priyansh Trivedi, and Jens Lehmann. "How to Revert Question Answering on Knowledge Graphs." In International Semantic Web Conference (Posters, Demos and Industry Tracks). 2017.

The methods used for KGQA dataset generation has been involving field expertise. In QALD Challenge, we see the researcher community publishes question with their corresponding sparql query. Here the size of the dataset is limited to a few hundred. WebQuestionSP [12] extends WebQuestions [14] by adding a formal query to the already existing question-answer pair. Following are the challenges faced while generating question answering over KG datasets:

1. A lot of manual effort is required.

2. Always required field expertise.

3. To phrase a complex question by a looking knowledge graph is difficult.

4. Understanding the KG Schema / Ontology is required.

As one of the pivotal requirements to evaluate and solve the QA problem is the availability of a large dataset comprising of varied questions and their logical forms, we will discuss it in details in Section 5.1. In this direction, we introduce the LC-QuAD (Large-Scale Complex Question Answering Dataset) dataset generation framework. We generated LC-QuAD 1.0 dataset consisting of 5000 questions along with the intended SPARQL queries required to answer questions over DBpedia. The dataset includes complex questions, i.e. questions in which the intended SPARQL query does not consist of a single triple pattern. We use the term "complex" to distinguish the dataset from simple questions corpus described in SimpleQuestions [10]. To the best of our knowledge, this is the largest QA dataset including complex questions with the next largest being Free917 [81] with 917 questions and QALD-6 [127] with 450 training questions and 100 test questions, respectively.

We frame our question generation problem as a transduction problem, similar to [11], in which KB *subgraphs* generated by the *seed entity* are fitted into a set of *SPARQL* templates which are then converted into a Normalized Natural Question Template (NNQT). This acts as a canonical structure which is then manually transformed into an NLQ having lexical and syntactic variations. Finally, a review is performed to increase the quality of the dataset.

The Chapter is organized into the following sections: (2) Relevance, where the importance of the resource is discussed; (3) Dataset Creation Workflow, where the approach of creating the dataset is discussed; (4) Dataset Characteristics; in which various statistics about the dataset are discussed; (5) Availability & Sustainability, describing the accessibility and long term preservation of the dataset; and (6) Conclusion & Future Work, summarizing and describing future possibilities.

5.1 Relevance

Relevance for Question Answering Research: Question answering approaches over structured data typically fall into two categories (as described in [128]): (i) *semantic parsing* based methods where the focus is to construct a semantic parser which can convert NLQs to an intermediate form, and then convert the intermediate form into a logical form, and (ii) *information retrieval* based techniques, which convert NLQs to a formal query language expression or directly to an answer, usually without any explicit intermediary form.

Approaches in the first category (semantic parsing based methods), frequently rely on handmade rules [15, 129]. Naturally, a goal of current research is to automate these manual steps. However, the size of the currently available training datasets is limited. The maximum size of the SPARQL-based QA dataset is 450 queries [127] and for λ-Calculus, the maximum size is 917 queries [81]. Due to these size limitations, it is currently unknown to what extent can these manual steps be automated. In particular, the relation between the size of a dataset, and the improvement in accuracy of employed ML techniques is unknown. The provision of LC-QuAD will allow to address these research questions in the future publication of semantic parsing based approaches.

Recent approaches in the second category (information retrieval based) are based on neural networks and have achieved promising results [10, 77]. However, these techniques are currently limited to answering simple questions, i.e. those which can be answered using a SPARQL query with a single triple pattern. Many queries are not simple: Comparative questions (e.g. "Was John Oliver born before Jon Stewart?"), Boolean questions (e.g. "Is Poland a part of Eurozone?"), questions involving fact aggregation (e.g. "Who has won the most Grammy awards?"), or even logically composite question (e.g. "In which university did both Christopher Manning and Sebastian Thrun teach?") cannot be answered by a system restricted to simple questions. We believe that it would be very interesting to explore neural network based approaches also for answering these complex questions. LC-QuAD provides initial foundations for exploring this research direction. While 5000 questions are likely insufficient in the long term, it should also be noted that the dataset size can be increased substantially by entity replacement (see Section 5.6). This dataset may enable neural networks based QA system to process a much larger variety of questions, and may lead to a substantial increase in their F-score.

Relevance for Other Research Areas

- **Entity and Predicate Linking**: During the expert intervention part of the workflow (see Section 5.2), the tokens referring to entities and predicates in the SPARQL query were edited as well. As a result, our dataset can be treated as a set of questions, along with a corresponding list of entities and predicates present in it. There are 5000 total questions, 615 predicates and 5042 entities in the dataset. In future work, we will release a version of the dataset where the questions are annotated with RDF entities.

- **SPARQL Verbalization**: This dataset can also assist the task of SPARQL verbalization, which has attracted research interest in the Semantic Web community [130, 131].

Relevance of and for the Semantic Web Community A significant portion of research in question answering over structured data has been done on non-RDF knowledge graphs [12, 77]. This could be attributed in part to the absence of large-scale QA datasets which use semantic technologies. By closing this gap via LC-QuAD, we believe that there can be a two fold benefit: On the one hand, researchers in question answering outside of the Semantic Web community can benefit from existing

W3C standards, such as SPARQL, as a framework for formalizing and approaching the QA problem. While, on the other hand, the Semantic Web community itself will be more centrally positioned in the area of question answering.

5.2 Dataset Generation Workflow

The primary objective while designing the framework for question generation was to generate a high quality large dataset with low domain expert intervention. In both QALD-6 [127], and Free917 [81], the logical forms of the questions were generated manually. This process of writing formal expressions needs domain experts with a deep understanding of the underlying KB schema, and syntaxes of the logical form. Secondly, following this approach makes the data more susceptible to human errors, as unlike natural language, formal languages are not fault tolerant.

5.2.1 Revert back KGQA Pipeline

To avoid these aforementioned shortcomings, instead of starting with NLQs and manually writing their corresponding logical forms, we invert the process. Figure 5.1 provides a outline of our dataset generation framework. KGQA systems translate NLQ to a formal query language expression, whereas in the LC-QuAD framework a formal query language expression and convert it to an NLQ. This reverse task is easier because formal query languages have well-defined semantics, and the entities and predicates occurring in the query are explicitly mentioned. Moreover, the target language (NL) is much more resilient to minute errors.

In the previous chapter, we explained how AskNowNQS converts an NLQ to an intermediate language (NQS) and then constructs the Formal (SPARQL) query In the LC-QuAD Framework, we reverse-engineered the architecture of AskNowNQS. We first generate SPARQL query, which is then translated to intermediate state language and then verbalized as Natural language question. Throughout the description, we will use the question: "Who is the wife of Barack Obama?" as our running example, to elaborate our dataset generation process in contrast to the process of answering this question. AskNowNQS breaks down the process of answering questions into two parts: conversion of questions into an expression of a semi-formal language, namely Normalized Query Structure (NQS) and subsequently converting the NQS expression to that of a formal language. NQS acts like a surface-level syntactic template for queries, which maps NLQs having different syntactic structures into a common structure. Given the question, AskNowNQS uses rules based on NL features to create a corresponding NQS instance. After that, it maps the entity (`dbr:Barack_Obama`), and the predicate (dbo:spouse) present in the question to resources in the target KB, i.e. DBpedia. After mapping, the NQS instance is converted into SPARQL using machine-generated, and hand made rules. Finally, the query is executed on the query endpoint of DBpedia to retrieve the required answer: `dbr:Barack_Obama`.

We begin our process where the solution of a KGQA problem such as AskNowNQS ends, i.e. by collecting the answer of the questions we aim to generate. For our framework, We take an entity, say "dbr:Michelle_Obama", and then consider this as the answer of the sparql. Now automatically generate a sparql query based on the subgraph around the selected entity, the sparql query is such that it returns as the entity chosen as the result. Next, the SPARQL is translated to question like structure, that works as an intermediate language. This intermediate language is an inverted version of the NQS discussed in the last chapter. Then, we manually write the natural language question based on the intermediate language and further verify it.

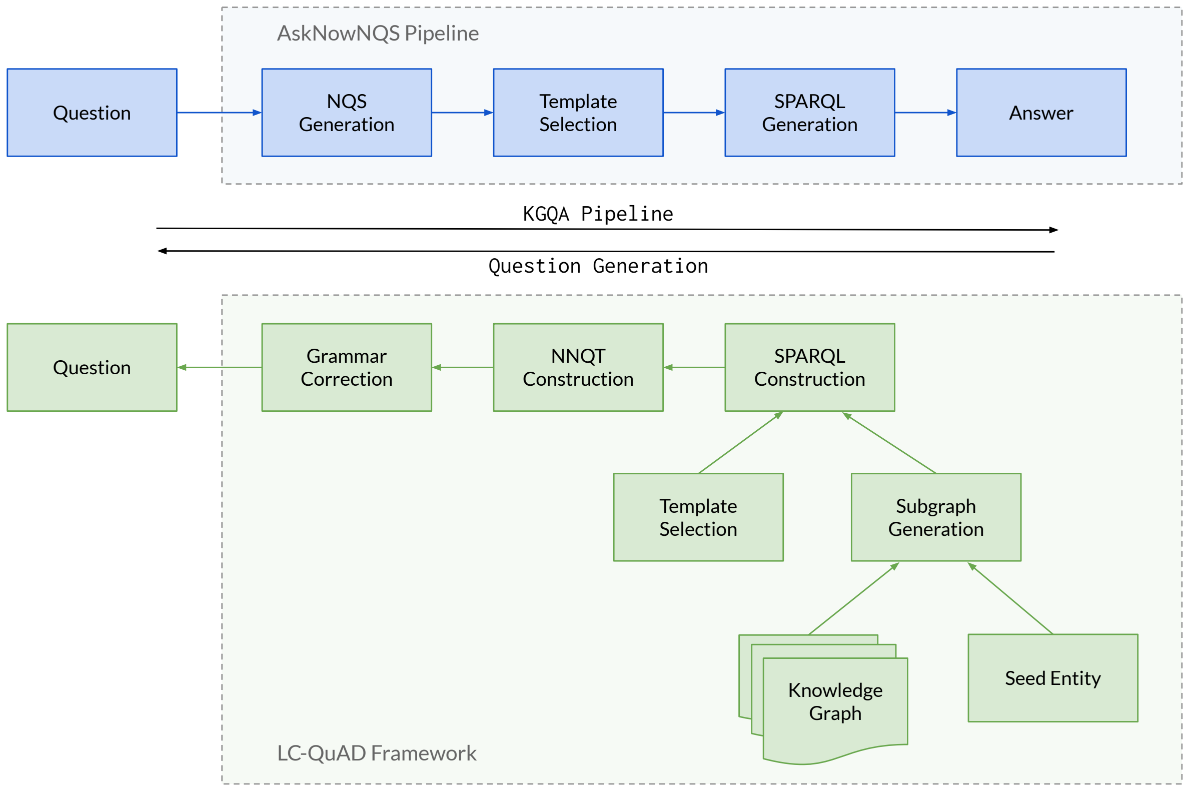


Figure 5.1: The Architecture of LC-QuAD Framework in Contrast to that of AskNowNQS

5.2.2 Formalization

In previous chapter we establish the formalization of KGQA system, where given a natural language question Q_{NL} we translate it to a formal language query Q_{FL} using an intermediate transitional state T_{state} such that

$$f_{kgqa} : Q_{NL} \mapsto T_{state} \mapsto Q_{FL}$$

Here in the scope of this chapter our aim is to inverse this function so that we can generate natural language question from formal language query. We establish this by again introducing intermediate state T^i, which holds the semantics of the formal language yet interpretable by non experts as it intact linguistic understanding. The protocol set here is to automatically generate formal query, map it to a intermediate state, final translate from this state to natural language question.

We could further formalize the LC-QuAD question generation framework as:.

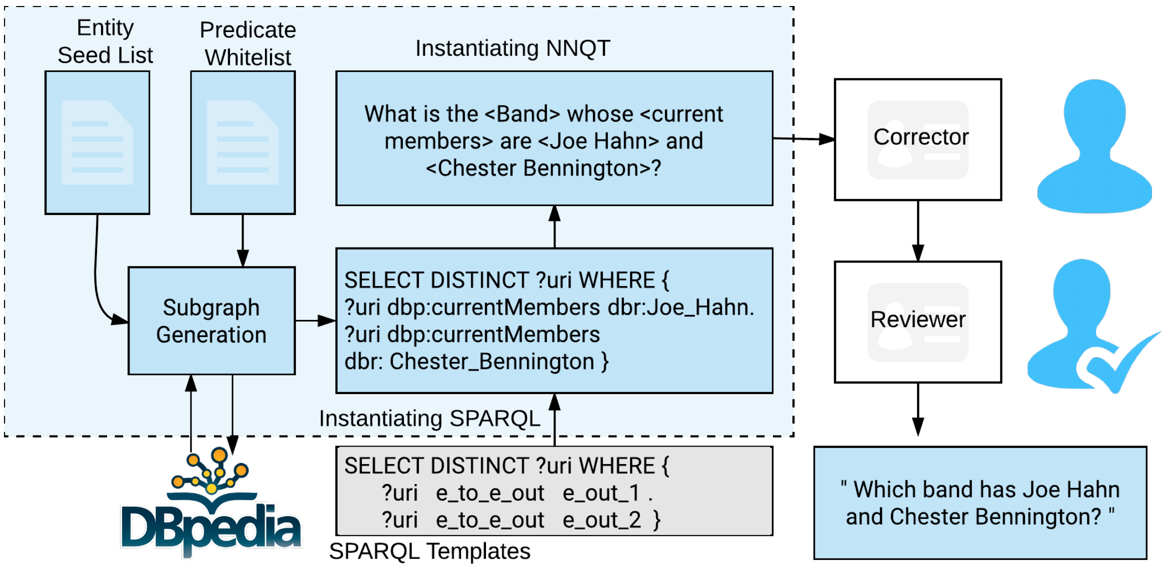

Entity Seed List
Predicate Whitelist
Instantiating NNQT
What is the <Band> whose <current members> are <Joe Hahn> and <Chester Bennington>?
Corrector
Subgraph Generation
SELECT DISTINCT ?uri WHERE { ?uri dbp:currentMembers dbr:Joe_Hahn. ?uri dbp:currentMembers dbr: Chester_Bennington }
Reviewer
Instantiating SPARQL
DBpedia
SELECT DISTINCT ?uri WHERE { ?uri e_to_e_out e_out_1 . ?uri e_to_e_out e_out_2 }
SPARQL Templates
" Which band has Joe Hahn and Chester Bennington? "

Figure 5.2: Using a list of seed entities, and filtering by a predicate white-list, we generate subgraphs of DBpedia to instantiate SPARQL templates, thereby generating valid SPARQL queries. These SPARQL queries are then used to instantiate NNQTs and generate questions (which are often grammatically incorrect). These questions are manually corrected and paraphrased by reviewers.

Formalization: LC-QuAD question generation

Given a formal language question Q_{FL} LC-QuAD question generation aims to model a transformation function f_{lcquad} such that :

$$f_{lcquad} : Q_{FL} \mapsto T^{i}_{state} \mapsto Q_{NL}$$

$$(Q_{FL})^{I_{FL}} \equiv (T_{state})^{I_{FL}, I_{NL}}$$

$$(T_{state})^{I_{FL}, I_{NL}} \equiv (Q_{FL})^{I_{NL}}$$

T_{state} is a intermediate transitional state holding semantic interpretation of the formal query Q_{FL} and understanding of Q_{NL}
$I_{FL}$is the semantic interpretation of the formal query.
I_{NL} is the linguistic Interpretation of the natural language question

5.2.3 Framework

We define the LC-QuAD framework in the following steps/modules:

Entity and Predicate Selection

One of the target of this dataset generation process is to generate question that a real user might pose to QA system. To achieve this target our question should be about known entities so that a good quality question could be framed. We manually pick entity from DBpedia such that the entities popular and we cover wide variety of entity type. We also manually picked informative predicates such that intuitive questions could be phrased and metadata predicate should be avoided. Our dataset is characteristic of the target KB, i.e. DBpedia. Thus, inconsistencies or semantically invalid triples in the KB can percolate into the dataset in the form of nonsensical questions. Since DBpedia has a lot of predicates which are used for metadata purposes, and are not of immediate semantic information[1], those should be avoided in the question generation process. To avoid these triples, we create a whitelist of 615 DBpedia predicates, and trim all the triples in the subgraph whose predicate is not in the whitelist. The list of seed entities[2], and a predicate white-list[3] are publicly available for figshare.com platform.

Sub-graph Generation

Then, for each entity in the list of seed entities, we extract subgraphs from DBpedia. Here, each subgraph contains triples within a 2-hop distance from the seed entity in the RDF graph. We clean the subgraph by removing the predicate (and the triple) which are not in our white-list of predicate. This allows us to have a clear subgraph, smaller in size and it allows us to have better coverage of white-list predicates in our SPARQL query, which are generated in subsequent steps.

Template Creation

We curated sparql templates from other QA datasets such as QALD and also further extend them based on heuristics. We finally had 35 templates covering single triple query, simple query with class information, two triple queries, ASK queries, COUNT queries and all with their variations. We fit the sparql-query-triple-pattern to the 2-hop subgraph generated in the previous stage. We discard all such query which gives null as a result. As shown in the figure, we have nomenclature within the subgraph. SPARQL templates and subgraph nomenclature are aligned, thus making it easy to generate sparql on the subgraph.

The previously described approach generates SPARQL queries with non-empty results over the target knowledge base. However, as human intervention is required to paraphrase each query into a question, we avoid generating similar questions. Herein, we define two questions to be similar if they have same SPARQL template, same predicates, and entities of same RDF class, which, when verbalized would also have a *similar* syntactic structure. For instance, Q1: *What is the capital of Germany?* has the following logical expression: `SELECT ?uri WHERE {dbr:Germany dbo:capital ?uri .}`. This question is similar to Q2: *What is the capital of France?* whose logical form is `SELECT ?uri WHERE {dbr:France dbo:capital ?uri .}`. Thus, in order to achieve more variations in our dataset with the same amount of human work, we prune the subgraphs to avoid generation of similar questions. In the future, we aim to automatically increase the size of our dataset by replacing entities (e.g. Germany in Q1) with entities of the same class (e.g. France in Q2).

[1]For e.g., dbo:abstract, dbo:soundRecording, dbo:thumbnail, dbo:wikiPageExternalLink, dbo:filename etc

[2]https://figshare.com/articles/Seed_Entities/5008286

[3]https://figshare.com/articles/White_List_Relations/5008283

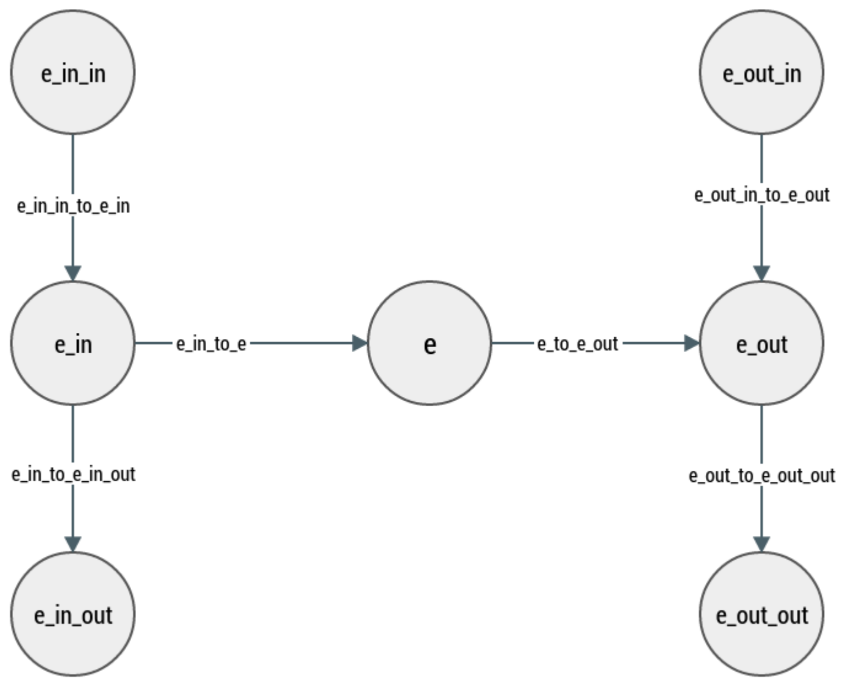


Figure 5.3: Nomenclature followed in Two-Hop Subgraph. Here ‘*e*‘ is the Central Entity which is Manually Picked

SPARQL to NNQT

Thereafter, we create an equivalent natural language template for every SPARQL template, called Normalized Natural Question Templates (NNQT). These are then instantiated to generate NLQs corresponding to every SPARQL query. The generated NLQs are often grammatically incorrect, but can be used by humans as a base for manual paraphrasing. The grammatical errors are due to fact that surface forms of DBpedia predicates correspond to varying parts of speech. For instance, while *president* is a noun, *largest city* is a modified noun, *bought* is a verb, whereas *born in* a prepositional phrase. These variations, along with complex entity surface forms (e.g. *2009 FIFA Club World Cup squads*) create a need for manual intervention to correct the grammar and paraphrase the questions.

NNQT to NLQ

This task can be done by fluent english speakers, who are not required to understand formal query languages, or the underlying schema of the KB. In this manner, using NNQT, we transduce the task of interpreting and verbalizing SPARQL queries, to a simpler task of grammar correction and paraphrasing, and thereby reduce the domain expertise required for our dataset generation process. We hosted a local server and provided User Interface so that this task could be done easily and reduce the hustle of handling the data. Our user interface showcased the SPARQL query and NNQT. The task given the ‘Corrector‘ is to correct the grammar of NNQT and also bring more variety in the question-style such as not to start every question with wh-word. Corrector is further provide the option to Report the question when some bug is noticed, and also Reject the instance if the question generated is not reasonable.

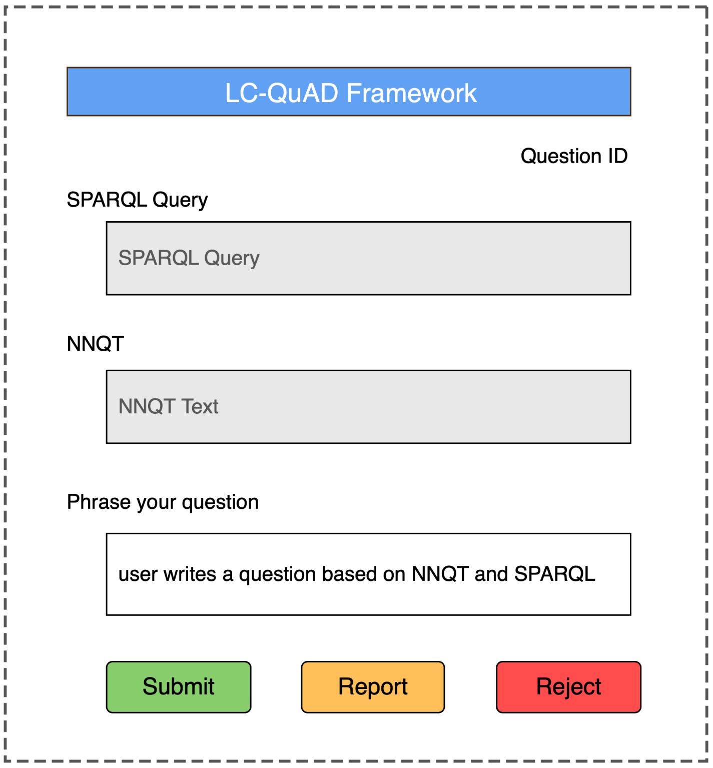


Figure 5.4: User Interface for Correcting NNQT to NLQ

Verification

Finally, every question is reviewed by an independent reviewer. This second iteration ensures a higher quality of data, since the reviewer is also allowed to edit the questions in case any errors are found.

.

5.3 Dataset Characteristics

Table 5.2 compares some statistics of QA Datasets over structured data. While QALD has 450 questions and Free917 has 917, LC-QuAD has 5000 questions. As mentioned in Section 5.1, QALD is the only dataset based on DBpedia, therefore, in this section we describe the characteristics of our dataset in contrast to it. Although LC-QuAD is tenfold in size compared to it, questions in QALD dataset are more complex and colloquial as they have been created directly by domain experts. Since the questions in our dataset are not extracted out of some external source, they are not an accurate representative of actual questions asked, but are characteristic of the knowledge base on which they were made. Nevertheless, due to human paraphrasing of both syntactic structure of the questions as well as the surface forms of entities and predicates, the questions in our dataset resemble questions actually asked by humans.

On an average, every question in our dataset has 12.29 tokens. The manual paraphrasing process was done by the native English speakers. Although the time taken to paraphrase a question varies significantly depending on the SPARQL template it is based on, it took about 48 seconds on average to correct each question. After this, the final reviewer took about 20 seconds to complete verification and, if needed, further editing. On the other hand, when a randomly sampled set of 100 SPARQL queries from our

Template	`SELECT DISTINCT ?uri WHERE { ?x  e_in_to_e_in_out e_in_out . ?x e_in_to_e ?uri }`
Query	`SELECT DISTINCT ?uri WHERE { ?x dbp:league dbr:Turkish_Handball_Super_League ?x dbp:mascot ?uri }`
NNQT Instance	What is the <mascot> of the <handball team> whose <league> is <Turkish Handball Super League >?
Question	What are the mascots of the teams participating in the turkish handball super league?
Template	`SELECT DISTINCT ?uri WHERE { ?x e_out_to_e_out_out e_out_out . ?uri e_to_e_out ?x }`
Query	`SELECT DISTINCT ?uri WHERE { ?x dbo:award dbr:BAFTA_Award_for_Best_Film_Music . ?uri dbo:musicComposer ?x }`
NNQT Instance	List the <movies> whose <music composer>'s <honorary title> is <BAFTA Award for Best Film Music>.?
Question	List down the movies whose music composers have won the BAFTA Award for Best Film Music ?

Table 5.1: Some Examples from LC-QuAD

dataset was given to the same people (without instantiated NNQTs), it took them about 94 seconds to verbalize a query. This indicates that our framework reduces the workload of creating QA datasets.[4]

Data Set	Size	Entities	Predicates	Formal Lang.
QALD-6	450	383	378	SPARQL
Free917	917	733	852	λ-Calculus
LC-QuAD	5000	5042	615	SPARQL

Table 5.2: A Comparison of Datasets having Questions and their Corresponding Logical Forms

Our dataset has 5042 entities and 615 predicates over 38 unique SPARQL templates. The SPARQL queries have been generated based on the 2016-04 DBpedia release[5]. Among the 5000 verbalized SPARQL queries, only 18% are simple questions, and the remaining queries either involve more than one triple, or COUNT/ASK keyword, or both. Moreover, we have 18.06% queries with a COUNT based aggregate, and 9.57% Boolean queries. As of now, we do not have queries with OPTIONAL, or UNION keyword in our dataset. Also, we do not have conditional aggregates in the query head.

[4]Naturally, the time required to start completely from scratch and think of a typical query and formalize it in SPARQL would be substantially higher and also lead to a low diversity from previous experience in the QALD challenge.

[5]http://wiki.dbpedia.org/downloads-2016-04

5.4 Availability and Sustainability

In this section, we describe the interfaces to access the dataset as well as how we plan to support sustainability. We have published our dataset on figshare[6] under CC BY 4.0[7] license. Figshare promises data persistence and public availability, thereby ensuring that the dataset should always be accessible regardless of the running status of our servers. The figshare project of LC-QuAD includes following files

- **LC-QuAD** - A JSON dump of Question Answering Dataset.
- **VoID description** - A machine readable description of the dataset in RDF.
- **Tertiary resources** - These include numerous resources, such as SPARQL templates, NNQTs, predicate white-lists etc. mentioned throughout the Chapter.

Regarding sustainability, the dataset will be integrated into the QALD challenge – specifically in QALD-8 and beyond. QALD is running since 2011 and recently the HOBBIT EU project has taken over its maintenance. From 2019 on, the HOBBIT association will run the challenge.

Our framework is available as an open source repository[8], under a GPL 3.0[9] License. The documentation of the framework, and its user manual have been published on the repository's Wiki as well. We intend to actively use Github issues to track feature requests and bug reports. Lastly, we will also announce all the new updates of the framework and dataset on all public Semantic Web lists.

5.5 Impact

In this section we look in to the impact of the LC-QuAD 1.0 dataset in KGQA research field. We also look into other extension to the dataset by us.

KGQA Baselines for Complex Questions

As LC-QuAD 1.0 is one of the first large scale complex KGQA dataset, specifically targeting DBpedia knowledge graph. With this dataset, there have been several approaches developed for tackling complex questions. QAmp [132], uses a message-passing based approach to solve complex questions. WDAqua [74] system does not perform any deep machine learning and instead handle complex question by n-gram technique and a handcrafted dictionary for predicate-matching. KrantikariQA [5] uses core chains with deep-learning and provide a more end-to-end solution once the question entity is available. We maintain the leader board for the KGQA performance over the LC-QuAD dataset at the LC-QuAD website `http://lc-quad.sda.tech/lcquad1.0.html`.

Fully Annotated LC-QuAD

KGQA relied on generic NLP components for various tasks, for example, entity linking. There are several popular entity linking systems available. Still, all of them are designed for large text and does not perform optimally on questions, where disambiguation is hard due to lack of context. With large scale dataset, now these QA components could be built explicitly for KGQA. We extended the LC-QuAD dataset with full annotation, which proved to be highly useful for KGQA components.

[6]https://figshare.com/projects/LC-QuAD/21812
[7]https://creativecommons.org/licenses/by/4.0/
[8]https://github.com/AskNowQA/LC-QuAD
[9]https://www.gnu.org/licenses/gpl.html

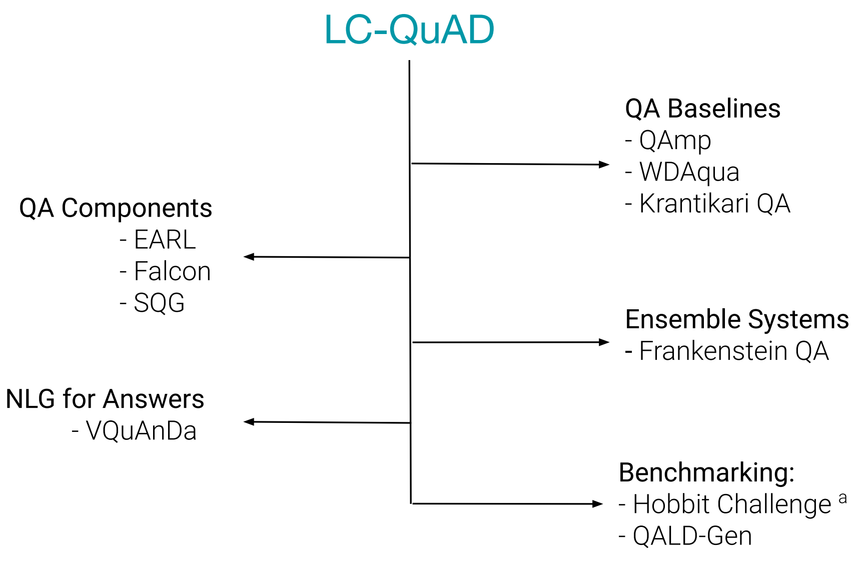


Figure 5.5: Impact of LC-QuAD 1.0 dataset

We first look in to the procedure followed during annotation process. We did a semiautomatic annotation over the whole dataset, where each question string was marked with entity and predicate present in the corresponding SPARQL. We first do a longest sub-string match with the entity label and mark the entity span. For relation span marking, we first deploy string matching over the question and predicate labels with synonyms. We also extended the predicate labels with the infection form of each word. Second last step was to use word embedding similarity between the predicated and every word in the question, marking the highest match as the relation span. In last, we performed a manual verification for this task. Thus a fully annotated version LC-QuAD 1.0 dataset is prepared. This dataset is available at figshare (link: `https://figshare.com/projects/EARL/28218`).

KGQA Components

As the dataset now have marked entity and relation span, it provided a good dataset for Entity linking and relation linking for KGQA. EARL [20] uses connection density in the subgraph to do entity and relation linking as joint task. Later followed by Falcon [89] which also did the join task with rule based mechanism. Further there are several other KGQA components which have been based on LC-QuAD 1.0 dataset. SQG [133] is a SPARQL Query Generator with a modular architecture using a ranking model based on Tree-LSTM. MDP [134] is distantly supervised learning framework based on reinforcement learning to learn the mentions of entities and relations in questions. FrankensteinQA [6] uses a ensemble approach to make dynamic question answering pipeline. VQuAnDa [115] is an answer verbalization dataset based on the LC-QuAD 1.0 answer set with several NLG baselines.

5.6 Conclusion

In this chapter, we described a framework for generating QA dataset having questions and their equivalent logical forms. This framework aims to reduce human intervention, thereby enabling the creation of larger datasets with fewer errors. We used it to create a dataset, LC-QuAD 1.0, having 5000 questions and their corresponding SPARQLs. Although we used DBpedia as the target KB for our dataset, the framework is KB agnostic. We compared the characteristics of the dataset with pre-existing datasets and also described its shortcomings.

We could also increase the number of SPARQL templates covered, thus increasing its syntactic variety. Moreover, to increase the size of the dataset by a certain factor, we can replace the entities in the questions with similar entities to synthetically add new questions. The software for this is already available and has been applied to create 2.1 million questions from 150 seed questions in QALD `https://github.com/hobbit-project/QuestionAnsweringBenchmark`. Increasing the dataset size in this way will likely benefit neural network based approaches for question answering as they learn the regularities in human language from scratch. However, this effect will diminish and estimating a factor up to which accuracy gains can be observed is a subject for future work. Additionally, a potential future research direction is to explore machine translation based techniques to reduce the need for manual grammar correction.

CHAPTER 6

A KGQA Data Set for Reified Knowledge Graphs

In the past decade knowledge graphs such as DBpedia[52] and Wikidata[95] have emerged as major successes by storing facts in linked data architecture. DBpedia recently decided to incorporate the manually curated knowledge base of Wikidata [135] into its own knowledge graph[1]. Retrieving factual information from these knowledge graphs has been a focal point of research. Question Answering over Knowledge Graphs (KGQA) is one of the techniques used to achieve this goal. In KGQA, the focus is generally on translating a natural language question to a formal language query. This task has generally been achieved by rule-based systems [15]. However, in the last few years, more systems using machine learning for this task have evolved. QA Systems have achieved impressive results working on simple questions [77] where a system only looks at a single fact consisting of a <subject - predicate - object> triple. On the other hand, for Complex questions (which require retrieval of answers based on more than one triple) there is still ample scope for improvement.

Datasets play an important role in AI research as they motivate the evolution of the current state of the art and the application of machine learning techniques that benefit from large-scale training data. In the area of KGQA, datasets such as WebQuestions, SimpleQuestions and the QALD challenge datasets have been the flag bearers. LC-QuAD version 1.0 was an important breakthrough as it was the largest complex question dataset using SPARQL queries at the time of its release.

Research Question 3 (RQ3)

How to incorporate Reification of Knowledge Graph in KGQA dataset so that the QA system utilize it?

The following are key contributions of this chapter:

- Provision of the largest dataset of 30,000 complex questions with corresponding SPARQL queries for Wikidata and DBpedia 2018.
- All questions in LC-QuAD 2.0 also consist of paraphrased versions via crowd-sourcing tasks. This provide more natural language variations for the question answering system to learn from and avoid over-fitting on a small set of syntactic variations.

[1]we refer this as 'DBpedia2018' further in this thesis.

- Questions in this dataset have a good variety and complexity levels such as multi-fact questions, temporal questions and questions that utilize qualifier information.
- This is the first KGQA dataset which contains questions with dual user intents and questions that require SPARQL string operations .

This chapter is based on the following publication ([17])

1. **Mohnish Dubey**, Debayan Banerjee, Abdelrahman Abdelkawi, and Jens Lehmann. "LC-QuAD 2.0 A large dataset for complex question answering over wikidata and dbpedia." In International Semantic Web Conference. Springer. 2019.

In this chapter, we present LC-QuAD 2.0 (Large-Scale Complex Question Answering Dataset 2.0) consisting of 30,000 questions with paraphrases and corresponding SPARQL queries required to answer questions over Wikidata and DBpedia2018. This dataset covers several new question type variations compared to the previous release of the dataset or to any other existing KGQA dataset (see comparison in Table 6.1). Apart from variations in the type of questions, we also paraphrase each question, which allows KGQA machine learning models to escape over-fitting to a particular syntax of questions. This is also the first dataset that utilizes qualifier [2] information for a fact in Wikidata, which allows a user to seek more detailed answers.

This Chapter is organized into the following sections: (2) Relevance and significance of the dataset and its possible impact (3) Dataset Creation Workflow (4) Dataset Characteristics in comparison with other KGQA datasets and look into the Availability and Sustainability of the dataset, and (5) Conclusion and Future Work.

6.1 Relevance

Question Answering: Over the last years, KGQA systems are trying to evolve from a handcrafted rule based system to more robust machine learning(ML) based systems. Such ML approaches require large datasets for training and testing. For simple questions the KGQA community has reached a high level of accuracy but for more complex questions there is scope for much improvement. A large scale dataset incorporates a high degree of variety in the formal query expressions. It provides a platform for machine learning models to improve the performance of KGQA with complex questions.

Solutions of NLP tasks using machine learning or semantic parsing have proved to be vulnerable to paraphrases. Moreover, if the system is exposed to paraphrases during the training period, the system could perform better and be more robust [136]. Thus, having paraphrases of each original question enlarges the scope of the dataset.

Recently, DBpedia decided adopting Wikidata's knowledge and mapping it to DBpedia's own ontology[135]. So far no datasets have based itself on this recent development. This work is the first attempt at allowing KGQA over the new DBpedia based on Wikidata [3].

Other Research Areas

Entity and Predicate Linking: This dataset may be used as a benchmark for systems which perform entity linking or/and relation linking on short text or on questions only. The previous version of the LC-QuAD

[2]Qualifiers are used in order to further describe or refine the value of a property given in a fact statement: `https://www.wikidata.org/wiki/Help:Qualifiers`

[3]at the time of writing this thesis, these updates do not reflect on the public DBpedia end-point. Authors have hosted a local endpoint of their own (using data from `http://downloads.dbpedia.org/repo/lts/wikidata/`). In future the authors shall release their own endpoint with new DBpedia model.

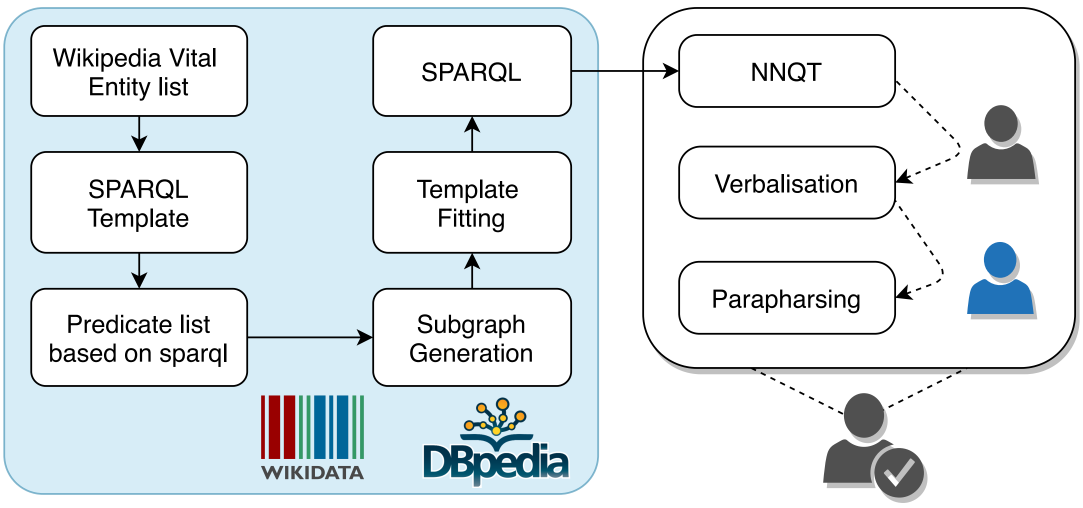


Figure 6.1: Workflow for the Dataset Generation

dataset has been used by such systems [20] and has enabled better performance of these modules.
SPARQL Query generation: The presented dataset has a high variety of SPARQL query templates which provides a use case for the modules which only focuses on generating SPARQL given a candidate set of entities and relations.The SQG system [133] uses tree LSTMs to learn SPARQL generation and used the previous version of LC-QuAD.
SPARQL to Natural language: This dataset may be used for natural language generation over knowledge graphs to generate complex questions at a much larger scale.

6.2 Dataset Generation Workflow

In this work, our aim is to generate different varieties of questions at a large scale. Although different kinds of SPARQLs are used the corresponding natural language questions generated need to appear coherent to humans. Amazon Mechanical Turk (AMT) was used for generating the natural language questions from the system generated templates. A secondary goal is to make sure that the process of verbalization of SPARQL queries on AMT does not require domain knowledge expertise of SPARQL and knowledge graphs on the part of the human workers (also known as turkers).

The core of the methodology is to generate SPARQL queries based on sparql templates, selected entities and suitable predicate. The SPARQLs are then transformed to Template Questions Q_T, which act as an intermediate stage between natural language and formal language. Then a large crowd sourcing experiment(AMT) is conducted where the Q_Ts are verbalized to natural language questions - i.e. verbalized questions Q_V and then later paraphrase them to the paraphrased questions Q_P. To clarify, a Q_T instance represents SPARQL in a canonical structure which is human understandable. The generation of Q_T is a rule based operation.

The workflow is shown in the Figure 6.1. The process starts with identifying a suitable set of entities for creating questions. A large set of entities based on Wikipedia Vital articles[4] is chosen and the corresponding same-as links to Wikidata IDs are found. Page-rank or entity popularity based approaches are avoided as it leads to dis-proportionately high number of entities from certain classes (say person). Instead Wikipedia Vital articles is chosen which provides important entities from a variety of topics such

[4]https://en.wikipedia.org/wiki/Wikipedia:Vital_articles/Level/5

as people, geography, arts and several more, along with sub-topics. As a running example, say "Barack Obama" is selected from the list of entities.

Next, a new set of SPARQL query templates are created such that they cover a large variety of question and intentions from a human perspective The template set is curated by observing other QA datasets and the KG architecture. All the templates have a corresponding SPARQL for Wikidata query end point and are valid on a DBpedia 2018 endpoint. The types of questions covered are as follows: simple questions (1 fact), multiple fact questions, questions that require additional information over a fact (wikidata qualifiers), temporal information questions, two intention questions and further discussed in Sec 4.3. Each class of questions also has multiple variations within the class.

Next, we select a predicate list based on the SPARQL template. For example if we want to make a "Count" question where user intends to know the number of times a particular predicate holds true, certain predicates such as "birthPlace" are disqualified as it will not make a coherent count-question. Thus different predicate white lists for different question types are maintained. Now the subgraph (Figure 6.2) is generated from the KG based on the three factors - entity ("Barack Obama"), SPARQL template (say two intentions with qualifier), and a suitable predicate list. After slotting the predicate and sub-graph into the template the final SPARQL is generated. This SPARQL is then transformed to natural language templates, henceforth known as Q_T (Question Template), and the process is taken over by three step AMT experiments as discussed further.

The First AMT Experiment - Here the aim is to crowd-source the work of verbalizing $Q_T \rightarrow Q_V$, where Q_V is the verbalization of Q_T performed by a turker. Note that Q_T, since system generated, is often grammatically incorrect and semantically incoherent, hence this step is required. For this we provided clear instruction to the turkers which vary according to the question type. For example: In two intention questions the turkers are instructed to make sure that none of the original intentions are missed in the verbalization. Sufficient number of examples are provided to turkers so that they understand the task well. Again the examples vary according to the question type in the experiment.

The Second AMT Experiment - The task given to the turkers was to paraphrase the questions which have been generated in experiment 1, $Q_V \rightarrow Q_P$, where Q_P is a paraphrase of Q_V such that Q_P preserves the overall semantic meaning of Q_V while changing the syntactic content and structure. Turkers are encouraged to use synonyms, aliases and further changing the grammar structure of the verbalized question.

The Third AMT Experiment - This experiment performs human verification of experiments 1 and 2 and enforces quality control in the overall work flow. Turkers compare Q_T with Q_V and also Q_V to Q_P, to decide if the two pairs carry the same semantic meaning.The turkers are given a choice between "Yes / No / Can't say".

6.3 Dataset Characteristics

6.3.1 Dataset Statistics

In this section we analyze the statistics of our dataset. LC-QuAD has 30,000 unique SPARQL - Question pairs. This dataset consists of 21,258 unique entities and 1,310 unique relations. Comparison of LC-QuAD 2.0 to other related datasets is shown in the Table 6.1. There are two datasets which cover simple questions, that is the question only requires one fact to answer. In this case the variation of formal queries is low. ComplexWebQuestion further extends the SPARQL of WebQuestions to generate complex questions. Though the number of questions in the dataset is in the same range as LC-QuAD 2.0, the variation of SPARQLs is higher in LC-QuAD 2.0 as it contains question 10 types question (such as boolean, dual intentions, Fact with qualifiers and other - ref 4.3) spread over 22 unique templates.

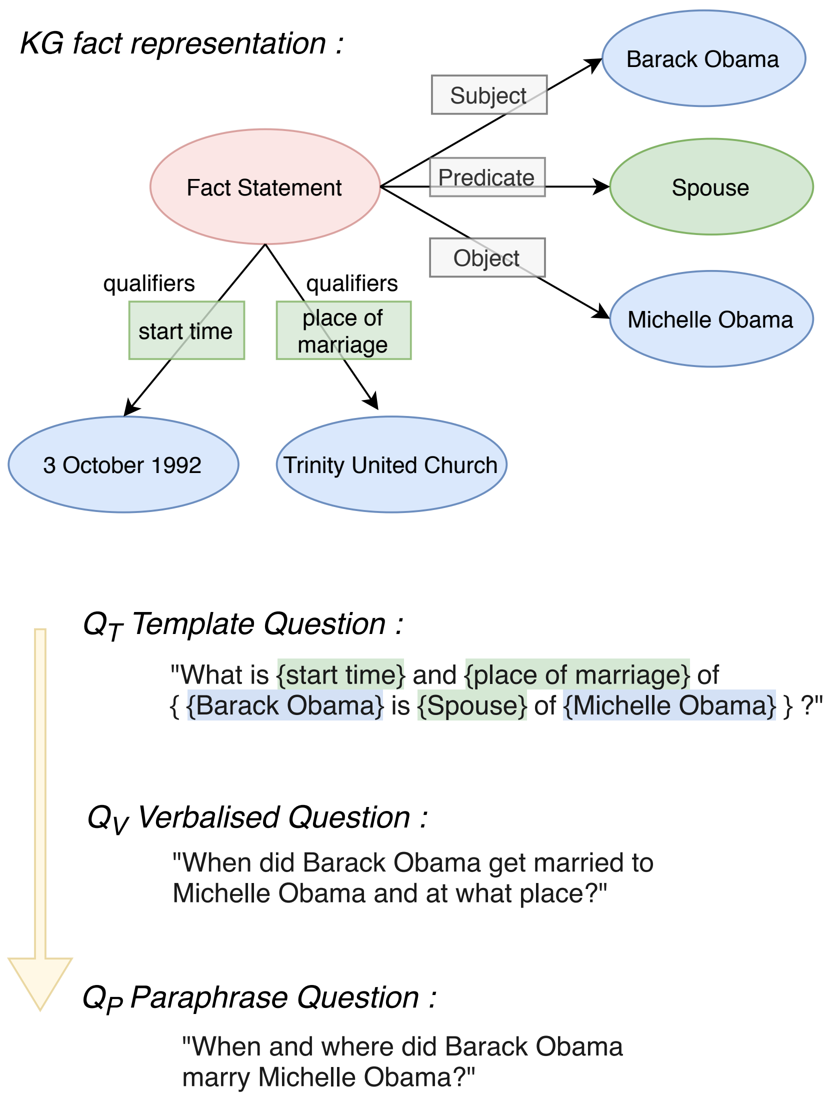

KG fact representation :
Barack Obama
Subject
Fact Statement
Predicate
Spouse
Object
Michelle Obama
qualifiers
start time
qualifiers
place of marriage
3 October 1992
Trinity United Church
Q_T Template Question :
"What is {start time} and {place of marriage} of { {Barack Obama} is {Spouse} of {Michelle Obama} } ?"
Q_V Verbalised Question :
"When did Barack Obama get married to Michelle Obama and at what place?"
Q_P Paraphrase Question :
"When and where did Barack Obama marry Michelle Obama?"

Figure 6.2: (top) Representation of a Fact with its Qualifiers. (bottom) Translation of a KG-fact to a Verbalized Question and then Paraphrased Question.

6.3.2 Analysis of Verbalization and Paraphrasing Experiments

To analyze the overall quality of verbalization and paraphrasing by turkers we also used some automated methods (see Figure 6.3). A good verbalization of a system generated template ($Q_T \rightarrow Q_V$) would mean that Q_V preserves the semantic meaning of Q_T with the addition and removal of certain words. However a good paraphrasing of this verbalization ($Q_V \rightarrow Q_P$) would mean that while the overall meaning is preserved, the order of words and also the words themselves (syntax) change to a certain degree. To quantify the sense of semantic-meaning vs change-of-word-order we calculate 1) cosine between vectors for each of these sentences pairs using BERT [137] embeddings - denoting "semantic similarity" 2) Levenshtein distance based syntax similarity between sentences showing the change in order of words.

We observe that the cosine similarities of Q_T, Q_V and Q_P stay high (mean between 0.8 - 0.9 with

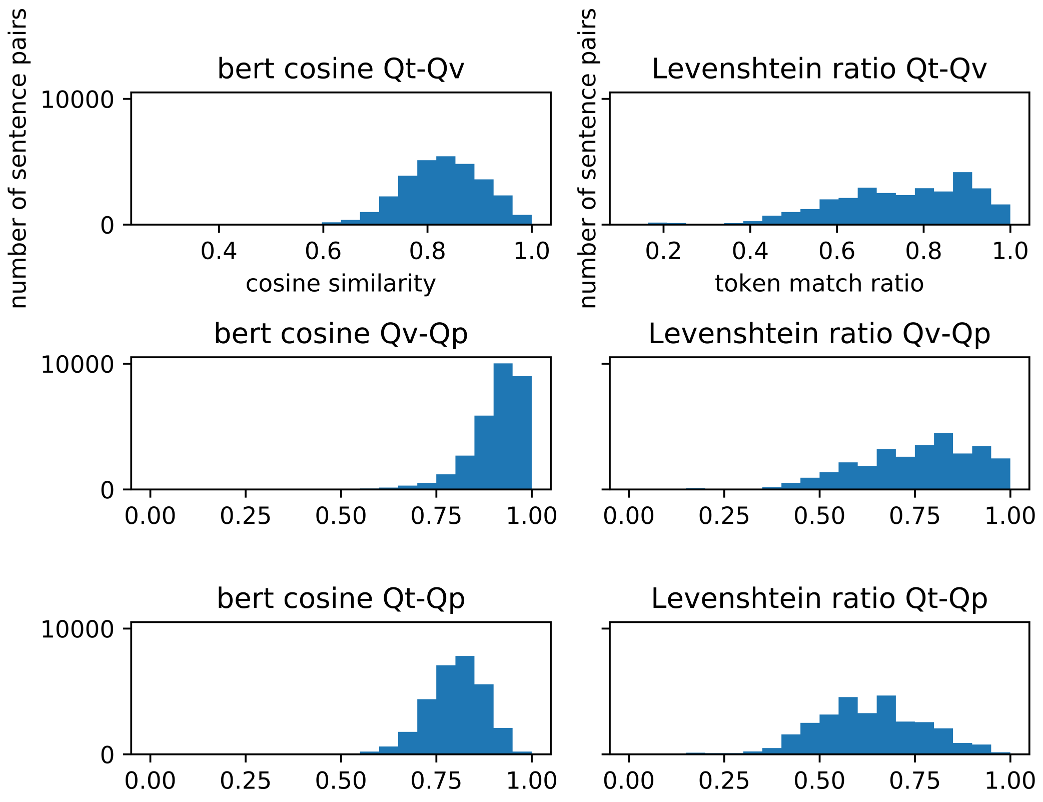


Figure 6.3: Comparing Q_T , Q_V, Q_P based on the parameter (a.)Semantic Similarity and (b.) Syntactic Similarity

Data Set	Size	Variation	Formal Lang.	Target KG	Paraphrase
Simple Questions[10]	100K	low	SPARQL	Freebase	No
30M Factoid Ques.[11]	30M	low	SPARQL	Freebase	No
QALD-9[13]	563	high	SPARQL	DBpedia	No
Free917[81]	917	medium	λ-Calculus	Freebase	No
WebQuestionSP[12]	5k	medium	SPARQL	Freebase	No
ComplexWebQues.[83]	34K	medium	SPARQL	Freebase	No
LC-QuAD 1.0 [16]	5k	medium	SPARQL	DBpedia 2016-04	No
LC-QuAD 2.0	30K	high	SPARQL	Wikidata & DBpedia2018	Yes

Table 6.1: A Comparison of Datasets having Questions and their Corresponding Logical Forms

standard deviation 0.07) denoting preservation of overall meaning throughout the steps, but syntax similarity stays comparatively low (mean between 0.6 - 0.75 with standard deviations between 0.14 to 0.16) since during verbalization several words are added and removed from the imperfect system generated templates, and during paraphrasing the very task is to change the order of words of Q_V

The last set of histograms shows semantic similarity between Q_T and Q_P directly. Since we have skipped the verbalization step in between we expect the distances to be farther away than other pairs. As expected the graphs show slightly lower cosine and syntax similarities than other pairs.

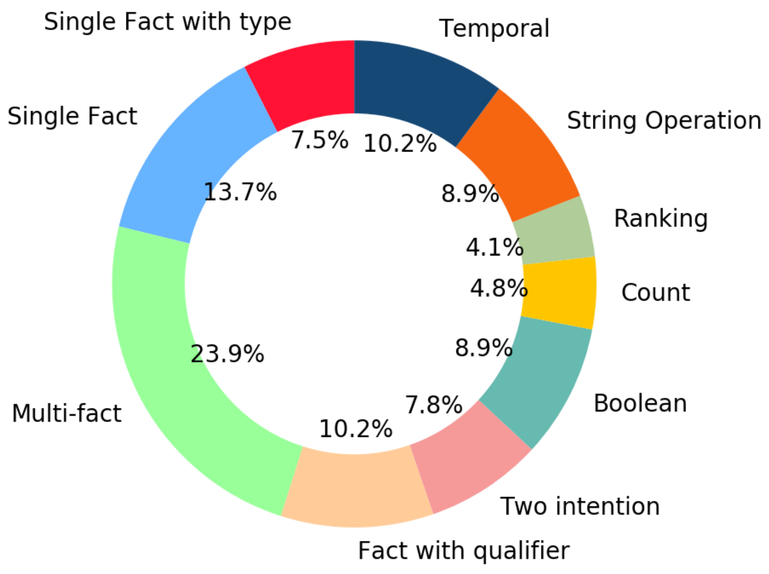


Figure 6.4: Distribution of Questions across all the Question Types

6.3.3 Availability and Sustainability

To support sustainability we have published the dataset at figshare under CC BY 4.010 license. URL: `https://figshare.com/projects/LCQuAD_2_0/62270`

The repository of LC-QuAD 2.0 includes following files
– LC-QuAD 2.0 - A JSON dump of the Question Answering Dataset. (Test and Train)
– The dataset is available with Template question Q_T, Question Q_V, paraphrased question Q_P and corresponding SPARQLs for Wikidata and DBpedia. Other supplementary material to the dataset can be accessed from our website `http://lc-quad.sda.tech/`

6.4 Variety in LC-QuAD 2.0

In this section we are look into the variety offered in the LC-QuAD 2.0 dataset. We are particularly looking in to the broad type of query in all the templates of this dataset are subsumed. We are showcasing the use of qualifier information sparql query, and how we are able to make more informative query by using the Reified Knowledge Graph. In each case we explain with examples stating the natural language question and the corresponding sparql query. The 6.4 gives the distribution of questions across all the question types published within the this dataset.

1. Single Fact

These queries are based on single fact (S-P-O). The two variations here are: one retrieving Object given the Subject and Predicate, other is retrieving Subject given the Predicate and Object. For example:
Question : "Who is the screenwriter of Mr. Bean?"
SPARQL : `select distinct ?answer where { wd:Q484020 wdt:P58 ?answer }`

2. Single fact with type

In these question we extend the Single triple pattern by adding "type of" constraint. The question generated in this template are informative compare to the question we discussed in single fact type question.
Question : "Billie Jean was on the tracklist of which studio album?"
SPARQL : `select distinct ?sbj where { ?sbj wdt:P658 wd:Q193319 .`
`?sbj wdt:P31 wd:Q208569 }`

3. Multi-fact

These queries are over two connected facts in Wikidata and have six variations to them.
Question : "What is the name of the sister city tied to Kansas City, which is located in the county of Seville Province?"
SPARQL : `SELECT ?answer WHERE { wd:Q41819 wdt:P190 ?answer .`
`?answer wdt:P1376 wd:Q95088 }`

4. Fact with qualifiers

As shown in the Figure 6.2, qualifiers are additional property for a fact stored in KG. LC-QuAD 2.0 utilize qualifiers to make more informative questions. For example, "What is the venue of Barack Obama's marriage ?" for this question we would use the qualifier 'place of marriage' to retrieve the answer.
Question: "What degree did Warren Buffet get at Columbia Business School?"
SPARQL: `SELECT ?value WHERE { wd:Q47213 p:P69 ?s .`
`?s ps:P69 wd:Q907481 .`
`?s pq:P512 ?value }`

5. Two intentions

This is a new category of query in KGQA, where the user question poses two intentions. This set of questions could also utilize the qualifier information as mentioned above and a two intention question could be generated, such as "Who is the wife of Barack Obama and where did he got married?" or "When and where did Barack Obama get married to Michelle Obama?" as highlighted in Figure 6.2.
Example: "Who is the film editor and director of Reservoir Dogs?"
SPARQL: `SELECT ?editor ?director WHERE`
`{ wd:Q72962 wdt:P1040 ?editor .`
`wd:Q72962 wdt:P57 ?director }`

6. Boolean

In Boolean question, user intends to know if the given fact is true or false. LC-QuAD 2.0 not only generates questions which returns true by graph matching, but also generate false facts so that Boolean question with "false" answers could be generated. We also use predicates that returns a number as an object, so that Boolean questions regarding numbers could be generated.
Question: "Is the electronegativity of the sodium less than 1.2?"
SPARQL: `ASK WHERE { wd:Q658 wdt:P1108 ?obj filter (?obj < 1.2) }`

Question: "Was it Hamlet that created the characters of Gertrude and The Ghost? "
SPARQL: `ASK WHERE { wd:Q41567 wdt:P674 wd:Q546616 .`
`wd:Q41567 wdt:P674 wd:Q1813659 }`

7. Count

This set of questions uses the keyword "COUNT" in SPARQL, and performs count over the number of times a certain predicate is used with a entity or object.
Question: "What is the number of Siblings of Edward III of England ?"
SPARQL: `SELECT ( COUNT (?obj) AS ?value )`
`{ wd:Q129247 wdt:P3373 ?obj }`

8. Ranking

By using aggregates, we generate queries where the user intends an entity with maximum or minimum value of a certain property. We have three variations in this set of questions.
Question:"what is the binary star which has the highest color index?"
SPARQL: `SELECT ?ent where { ?ent wdt:P31 wd:Q50053.`
`?ent wdt:P1458 ?obj}`
`ORDER BY DESC (?obj) LIMIT 5`

9. String Operation

By applying string operations in SPARQL we generated questions where the user asks about an entity either at word level or character level.
Question: "WHICH IS THE ROCK BAND THAT BEGINS WITH Y"
SPARQL: `SELECT DISTINCT ?sbj ?sbjlabel WHERE`
```
{ ?sbj wdt:P31 wd:Q5741069 .  ?sbj rdfs:label ?sbjlabel .
FILTER (STRSTARTS (lcase (?sbjlabel), 'y') ) .
FILTER (lang (?sbjlabel) = 'en') } LIMIT 25
```

Question : "Tell me city whose name has the word whitehorse in it."
SPARQL: `SELECT DISTINCT ?sbj ?sbjlabel WHERE`
```
{ ?sbj wdt:P31 wd:Q515 .  ?sbj rdfs:label ?sbjlabel .
FILTER (CONTAINS (lcase (?sbjlabel), 'whitehorse') ) .
FILTER (lang (?sbjlabel) = 'en') } LIMIT 25
```

10. Temporal aspect

This dataset covers temporal property in the question space and also in the answer space. A lot of the times facts with qualifiers poses temporal information.
Question: "What prize was William Butler Yeats shortlisted for in year 1918?"
SPARQL: `SELECT ?obj WHERE { wd:Q40213 p:P1411 ?s .`
```
?s ps:P1411 ?obj .
?s pq:P585 ?x
filter (contains (YEAR(?x),'1918')) }
```

6.5 Conclusion

In this Chapter, we presented LC-QuAD 2.0 the First large scale dataset on Reified Knowledge Graphs. We extended the LC-QuAD Framework to generate the dataset in a semi-automatic setting that further requires crowd-sourcing stages without domain knowledge expertise. LC-QuAD 2.0 has a lot more variation compared to any KGQA dataset currently present. The few new varieties introduced in this dataset are questions with temporal reasoning, two intentions, multi-fact, string operations and KG-qualifier based questions. We extended Boolean questions with mathematical operations such as 'greater than', 'smaller than' and 'equals to'. We generate these questions by creating sparql queries and then perform a template-based translation to NNQT, which represent a natural language like structure. We set up a three crowd-sourcing experiments, first, to correct the NNQT based questions, experiment two is to paraphrase the question from the previous step, and finally perform an inter-annotator agreement on the output of the last two experiments. We introduced paraphrasing of the question in this iteration of LC-QuAD dataset to deviate further from the templates and having more syntactic diversity in the questions. Potential future work is of developing a baseline KGQA system using the LC-QuAD 2.0 dataset. A benchmark leader-board for KGQA systems on this dataset is maintained on our website. In future we can fully automate the process of question generation by using Natural Language Generation techniques.

CHAPTER 7

Joint Entity and Relation Linking for KGQA

In Chapter 4, we focused on building KGQA system based on intermediate language. There were two significant research gaps highlighted post the system development. One shortcoming was the lack of large scale KGQA dataset with complex questions. In Chapter 5 and Chapter 6, we worked towards generating large scale KGQA dataset with a semi-automatic process. The second shortcoming of the KGQA systems was the reliance on NLP component that are not mainly designed for question answering task. Most crucial components are Entity linking and relation linking that are not intended for question answering settings. The bad performance of these fundamental components always hampers the performance of the entire QA system. With the success of generating large scale KGQA dataset, the community could look forward to design NLP component specific to KGQA. Thus, creating Entity Linking and Relation Linking components for KGQA pipeline is now achievable.

The performance of existing Entity Linker is useful on large text data, but the accuracy significantly drops when we test over question strings. The cause of this dip of accuracy is the lack of context the question strings offer as compared to a long text document. A question would usually have just one or two entity whereas a large text would have many more entities. Thus, judging the context becomes easier in long texts. Similarly, Relation Linking is a significant step in question answering and offers a lot of challenges. Relation Linking and Entity Linking have common goals, i.e. to identify the spans in the text and connect them to the knowledge graph. Further in literature, we have seen systems improving entity linking accuracy by using relation linking output and vice-versa. In this Chapter, we focus on performing entity linking and relation linking together as one task.

Research Question 4 (RQ4)

How effective is the joint task of Entity Linking task and Relation Linking for KGQA?

Overall, our contributions in this Chapter are as follows:

- The framework EARL, where GTSP solver or Connection Density can be used for joint linking of entities and relations (Sec. 7.2).
- A formalization of the joint entity and relation linking problem as an instance of the Generalised Travelling Salesman (GTSP) problem (Sec. 7.2.2).
- An implementation of the GTSP strategy using approximate GTSP solvers.

- A "Connection Density" formalization and implementation of the joint entity and relation linking problem as a machine learning task (Sec. 7.2.3).
- An adaptive E/R learning module, which can correct errors occurring across different modules (Sec. 7.2.4).
- A comparative analysis of both strategies - GTSP and connection density (Table 7.2).
- A fully annotated version of the 5000 question LC-QuAD data-set, where entity and relations are linked to the KG.
- A large set of labels for DBpedia predicates and entities covering the syntactic and semantic variations.[1]

This chapter is based on the following publications([20, 138])

1. **Mohnish Dubey**, Debayan Banerjee, Debanjan Chaudhuri, and Jens Lehmann. "Earl: Joint entity and relation linking for question answering over knowledge graphs." In International Semantic Web Conference, pp. 108-126. Springer, Cham, 2018.
2. Debayan Banerjee, **Mohnish Dubey**, Debanjan Chaudhuri, and Jens Lehmann. "Joint Entity and Relation Linking Using EARL." In International Semantic Web Conference (P&D/Industry/BlueSky). 2018.

In this chapter we develop a methodology to create module which performs entity linking and relation linking as common joint task. Traditionally the EL and RL tasks have been performed as individual tasks in KGQA. These two tasks are either performed as parallel task or as sequential task. In parallel setting both the tasks do not share any information, hence lack in local context. In sequential task settings the error in one task cascades to the other task. We propose EARL ($\underline{E}$ntity $\underline{a}$nd $\underline{R}$elation $\underline{L}$inker), a system for jointly linking entities and relations in a question to a knowledge graph. EARL treats entity linking and relation linking as a single task and thus aims to reduce the error caused by the dependent steps.

Usually, all entities and relations need to be correctly linked to the knowledge graph in order to generate the correct formal query and successfully answer the question of a user. Hence, it is crucial to perform the linking process with high accuracy and this is a major bottleneck for the widespread adoption of current SQA systems. In most entity linking systems [18, 19], disambiguation is performed by looking at other entities present in the input text. However, in the case of natural language questions (short text fragments) the number of other entities for disambiguation is not high. Therefore, it is potentially beneficial to consider entity and relation candidates for the input questions in combination, to maximize the usable evidence for the candidate selection process. To achieve this, we propose EARL ($\underline{E}$ntity $\underline{a}$nd $\underline{R}$elation $\underline{L}$inker), a system for performing entity linking and relation linking in a question to a knowledge graph as a joint task.

EARL uses the knowledge graph to jointly disambiguate entity and relations: It obtains the context for entity disambiguation by observing the relations surrounding the entity. Similarly, it obtains the context for relation disambiguation by looking at the surrounding entities. The system supports multiple entities and relations occurring in complex questions. EARL implements two different solution strategies: The first strategy is a formalization of the joint entity and relation linking tasks as an instance of the Generalised Travelling Salesman Problem (GTSP). Since the problem is NP-hard, we employ approximate GTSP solvers. The second strategy uses machine learning in order to exploit the connection density

[1]dataset available at `https://github.com/AskNowQA/EARL`

between nodes in the KG. It relies on three base features and re-ranking steps in order to predict entities and relations. We compare the strategies and evaluate them on a dataset with 5000 questions. Both strategies outperform the current state-of-the-art approaches for entity and relation linking.

Let us consider an example to explain the underlying idea: *"Where was the founder of Tesla and SpaceX born?"*. Here, the entity linker needs to perform disambiguation for the keyword "Tesla" between the scientist "Nikola Tesla" and the car company "Tesla Motors". EARL uses all other entities and relations (*SpaceX, founder, born*) present in the query. It does this by analyzing the subdivision graph of the knowledge graph fragment containing the candidates for relevant entities and relations. While performing the joint analysis (Figure 7.1), EARL detects that there is no likely combination of candidates, which supports the disambiguation of "Tesla" as "Nikola Tesla", whereas there is a plausible combination of candidates for the car company "Tesla Motors".

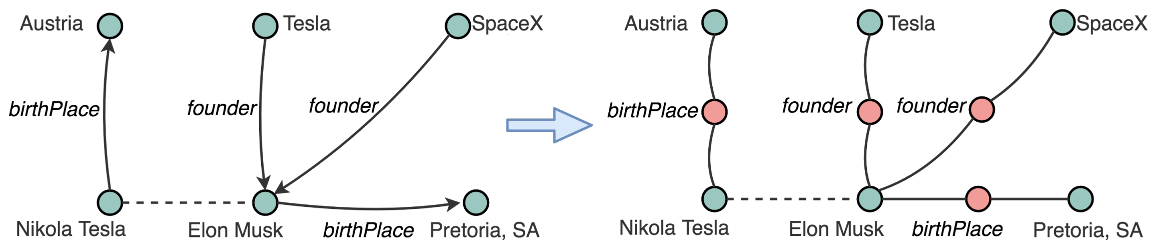


Figure 7.1: An excerpt of the Subdivision Knowledge Graph for the example question "Where was the founder of Tesla and Space X born?". Note that both Entities and Relations are Nodes in the Graph.

The Chapter is organized into the following sections:

(2) Problem Statement, where we discuss the problem in depth and our hypotheses for the solution; (3) the architecture of EARL including preprocessing steps followed by (i) a GTSP solver or (ii) a connection density approach; (4) Availability of EARL demo and api in public; (5) Evaluation, with various evaluation criteria and results; (6) Discussion; and (7) Conclusion.

7.1 Overview and Preliminaries

7.1.1 Overview and Research Questions

As discussed previously, in question answering the tasks of entity and relation linking are performed either sequentially or in parallel. In sequential systems, usually the entity linking task is performed first, followed by relation linking. As a consequence, information in the relation linking phase cannot be exploited during entity linking in this case. In parallel systems, entity and relation linking are performed independently. While this is efficient in terms of run-time performance, the entity linking process cannot benefit from further information obtained during relation linking and vice versa. We illustrate the advantages and disadvantages of both approaches, as well as the systems following them, in Table 7.1. Our main contribution in this Chapter is the provision of a system, which takes candidates for entity and relation linking as input and performs a joint optimization selecting the best combination of entity and relation candidates.

Postulates We have three postulates, which we want to verify based on our approach:

H1: Given candidate lists of entities and relations from a question, the correct solution is a cycle of minimal cost that visits exactly one candidate from each list.

Linking Approach	QA System	Advantage	Disadvantage
Sequential	[15] [139][6]	-Reduces candidate search space for Relation Linking -Allows schema verification	-Relation Linking information cannot be exploited in Entity Linking process - Errors in Entity Linking cannot be overcome
Parallel	[140] [106][141]	- Lower runtime - Re-ranking of Entities possible based on Relation Linking	- Entity Linking process cannot use information from Relation Linking process and vice versa - Does not allow schema verification
Joint (with limited candidate set)	[14] [78]	- Potentially high accuracy - Reduces error propagation - Better disambiguation - Allows schema verification - Allows re-ranking	- Complexity increase - Larger search space

Table 7.1: State of the art for Entity and Relation linking in Question Answering

H2: Given candidate lists of entities and relations from a question, the correct candidates exhibit relatively dense and short-hop connections among themselves in the knowledge graph compared to wrong candidate sets.

H3: Jointly linking entity and relation leads to higher accuracy compared to performing these tasks separately.

We will re-visit all of these postulates in the evaluation section of the Chapter.

7.1.2 Preliminaries

We will first introduce basic notions from graph theory:

Definition 7.1.1 (Subdivision Graph) *The subdivision graph [142] $S(G)$ of a graph G is the graph obtained from G by replacing each edge $e = (u, v)$ of G by a new vertex w_e and 2 new edges (u, w_e) and (v, w_e) .*

7.2 EARL

In general, entity linking is a two step process. The first step is to identify and spot the span of the entity. The second step is to disambiguate or link the entity to the knowledge graph. For linking, the candidates are generated for the spotted span of the entity and then the best candidate is chosen for the linking. These two steps are similarly followed in standard relation linking approaches. In our approach, we first spot the spans of entities and relations. After that, the (disambiguation) linking task is performed jointly for both entities and relations.

In this section we first discuss the step of span detection of entity and relation in natural language question and candidate list generation. We perform the disambiguation by two different approaches, which are discussed later in this section.

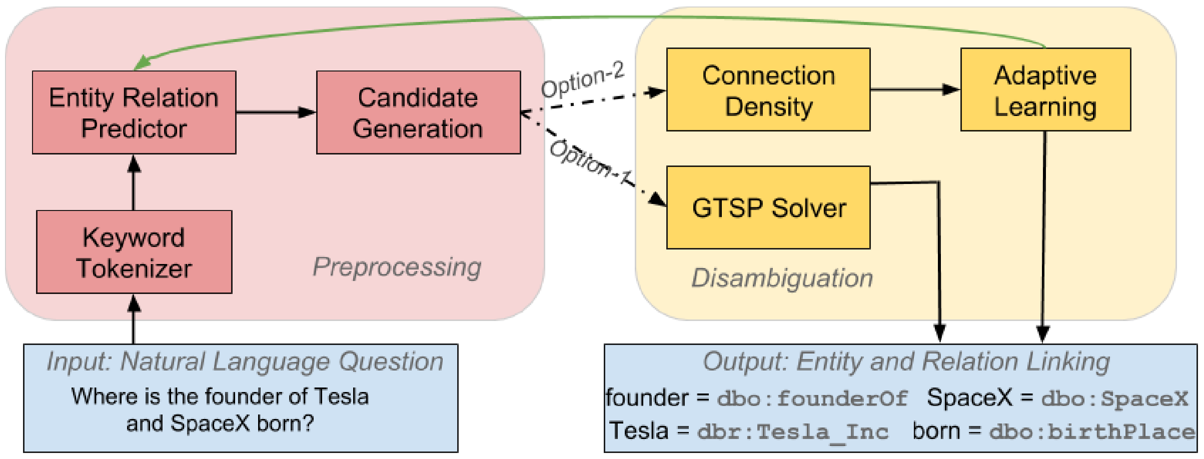

Entity Relation Predictor
Candidate Generation
Keyword Tokenizer
Preprocessing
Option-2
Option-1
Connection Density
Adaptive Learning
GTSP Solver
Disambiguation
Input: Natural Language Question
Where is the founder of Tesla and SpaceX born?
Output: Entity and Relation Linking
founder = dbo:founderOf SpaceX = dbo:SpaceX
Tesla = dbr:Tesla_Inc born = dbo:birthPlace

Figure 7.2: EARL Architecture: In the disambiguation phase one may choose either Connection Density or GTSP. In cases where training data is not available beforehand GTSP works better.

7.2.1 Candidate Generation Steps

Shallow Parsing

Given a question, extract all keyword phrases out. EARL uses SENNA[143] as the keyword extractor. We also remove stop words from the question at this stage. In example question "Where was the founder of Tesla and SpaceX born?" we identify <*founder, Tesla, SpaceX, born*> as our keyword phrases.

E/R Prediction

Once keyword phrases are extracted from the questions, the next step in EARL is to predict whether each of these is an entity or a relation. We use a character embedding based long-short term memory network (LSTM) to do the same. The network is trained using labels for entity and relation in the knowledge graph. For handling out of vocabulary words [144], and also to encode the knowledge graph structure in the network, we take a multi-task learning approach with hard parameter sharing. Our model is trained on a custom loss given by:

$$\mathcal{E} = (1 - \alpha) * \mathcal{E}_{BCE} + \alpha * \mathcal{E}_{ED} \tag{7.1}$$

Where, $\mathcal{E}_{BCE}$ is the binary cross entropy loss for the learning objective of a phrase being an entity or a relation and $\mathcal{E}_{Ed}$ is the squared eucledian distance between the predicted embedding and the correct embedding for that label. The value of α is empirically selected as 0.25. We use pre-trained label embeddings from RDF2Vec [145] which are trained on knowledge graphs. RDF2Vec provides latent representation for entities and relations in RDF graphs. It efficiently captures the semantic relatedness between entities and relations.

We use a hidden layer size of 128 for the LSTM, followed by two dense layers of sizes 512 and 256 respectively. A dropout value of 0.5 is used in the dense layers. The network is trained using Adam optimizer [146] with a learning rate of 0.0001 and a batch size of 128. Going back to the example, this module identifies "founder" and "born" as *relations*, "Tesla" and "SpaceX" as *entities*.

Candidate List Generation

This module retrieves a candidate list for each keyword identified in the natural language question by the shallow parser. To retrieve the top candidates for a keyword we create an Elasticsearch[2] index of URI-label pairs. Since EARL requires an exhaustive list of labels for a URI in the knowledge graph, we expand the labels. We used Wikidata labels for entities which are in same-as relation in the knowledge base. For relations we require labels which were semantically equivalent (such as writer, author) for which we took synonyms from the Oxford Dictionary API [3]. To cover grammatical variations of a particular label, we added inflections from fastText[4]. We avoid any bias held towards or against popular entities and relations.

The output of these pre-processing steps are i) set of keywords from the question, ii) every keyword is identified either as relation or entity, iii) for every keyword there is a set of candidate URIs from the knowledge graph.

Creating an Extended Label Vocabulary

In the text search phase we need to retrieve a candidate list for each keyword identified in the natural language question by the shallow parser. To retrieve the top candidates for a keyword we create an Elasticsearch index of uri-label pairs. Since EARL requires exhaustive list of labels for a DBpedia uri we expanded the labels beyond the dbpedia provided label. We used Wikidata labels using dbpedia "same-as" links for entities. For example label for dbr:BarackObama is only "Barack Obama" in DBpedia, but we expand this set by using Wikidata sameas, thus our index has labels "Barack Obama, Barack Hussein Obama, President Obama, Barack, ... ". For relations we required labels which were semantically equivalent for which we took synonyms from the Oxford Dictionary API. For example dbo:writer has the label "writer" in DBpedia; using Oxford-Dictionary we expand with labels such as "author, penman, creator, ...". EARL further uses FastText for covering all the inflection forms of these labels, such "write, writer, writing, wrote, written, ...". Our extended vocabulary contains 256K labels for DBpedia relations. We disable the default TF-IDF based scoring so that no bias is held towards or against popular entities and relations.

Using Bloom filters for fast querying of the KB

While performing joint linking EARL checks connectivity and distance between two nodes of the knowledge graph. EARL checks the connection between a candidate of a keyword phrase to all the candidates of all the other keyword phrases. Two DBpedia nodes may have multiple intermediate nodes in the path connecting them and looking at all of them takes a large amount of time as the number of such (pair of nodes) queries are high. We do not directly query the knowledge graph as the execution time for such a query in a large knowledge graph is prohibitively high.
EARL uses *Bloom filters* [147], that are probabilistic data structures which can answer questions of set membership, like "is-a-member-or-not" in constant O(1) time. The user can decide the acceptable error rate when creating the Bloom filter by choosing the appropriate size of the Bloom filter and corresponding number of hash functions. EARL uses Bloom filter parameters so that it has 1 in a million error probability.
There are separate Bloom filters for different hop length pairs. We took each such pair as a string (eg: *http://dbpedia.org/resource/car:http://dbpedia.org/resource/bike*) and added it to the corresponding

[2] `https://www.elastic.co/products/elasticsearch`
[3] `https://developer.oxforddictionaries.com/`
[4] `https://fasttext.cc/`

Bloom filter. Hence we ended up with a 4 Bloom filter with different lengths (upto 4hops), each around 1 GB in size resident in-memory. This method allows us to condense a KG which is several hundred GBs in size into only a few GBs with a low probability of error. When we need to find connection density, we take each pair of candidate URIs from the lists returned by Elasticsearch and ask the Bloom filters if they contain these URI pairs or not. Depending on which Bloom filter answered positively we know how many hops away these two URIs are in the knowledge graph.

7.2.2 Using GTSP for Disambiguation

At this point we may use either a GTSP based solution or Connection Density (later explained in 4.3) for disambiguation. We start with the formalization for GTSP based solution.

The entity and relation linking process can be formalized via spotting and candidate generation functions as follows: Let S be the set of all strings. We assume that there is a function $spot : S \rightarrow 2^S$ which maps a string s (the input question) to a set $\mathcal{K}$ of sub-strings of s. We call this set $\mathcal{K}$ the *keywords* occurring in our input. Moreover, we assume there is a function $\text{cand}_{KG} : \mathcal{K} \rightarrow 2^{V \cup L}$ which maps each keyword to a set of candidate node and edge labels for our knowledge graph $G = (V, E, L)$. The goal of joint entity and relation linking is to find combinations of candidates, which are closely related. How closely nodes are related is modelled by a cost function $\text{cost}_{KG} : (V \cup L) \times (V \cup L) \rightarrow [0, 1]$. Lower values indicate closer relationships. According to our first postulate, we aim to encode graph distances in the cost function to reward those combinations of entities and relations, which are located close to each other in the input knowledge graph. To be able to consider distances between both relations and entities, we transform the knowledge graph into its subdivision graph (see Definition 7.1.1). This subdivision graph allows us to elegantly define the distance function as illustrated in Figure 7.7.

Given the knowledge graph KG and the functions $spot$, $cand$ and $cost$, we can cast the problem of joint entity and relation linking as an instance of the Generalized Travelling Salesman (GTSP) problem: We construct a graph G with $V = \bigcup_{k \in K} \text{cand}(k)$. Each node set cand($k$) is called a cluster in this vertex set. The GTSP problem is to find a subset $V' = (v_1, \ldots, v_n)$ of V which contains exactly one node from each cluster and the total cost $\sum_{i=1}^{n-1} \text{cost}(v_i, v_{i+1})$ is minimal with respect to all such subsets. Please note that in our formalization of the GTSP, we do not require V' to be a cycle, i.e. v_1 and v_n can be different. Moreover, we do not require clusters to be disjoint, i.e. different keywords can have overlapping candidate sets.

Figure 7.3 illustrates the problem formulation. Each candidate set for a keyword forms a cluster in the graph. The weight of each edge in this graph is given by the cost function, which includes the distance between the nodes in the subdivision graph of the input knowledge graph as well as the confidence scores of the candidates. The GTSP requires the solution to visit one element per cluster and minimises the overall distance.

Approximate GTSP Solvers

In order to solve the joint entity and relation linking problem, the corresponding GTSP instance needs to be solved. Unfortunately, the GTSP is NP-hard [148] and hence it is intractable. However, since GTSP can be reduced to standard TSP, several polynomial approximation algorithms exist to solve GTSP. The state-of-the-art approximate GTSP solver is the Lin–Kernighan–Helsgaun algorithm [149]. Here, a GTSP instance is transformed into standard asymmetric TSP instances using the Noon-Bean transformation. It allows the heuristic TSP solver LKH to be used for solving the initial GTSP. Among LKH's characteristics, its use of 1-tree approximation for determining a candidate edge set, the extension of the basic search step, and effective rules for directing and pruning the search contribute to its efficiency.

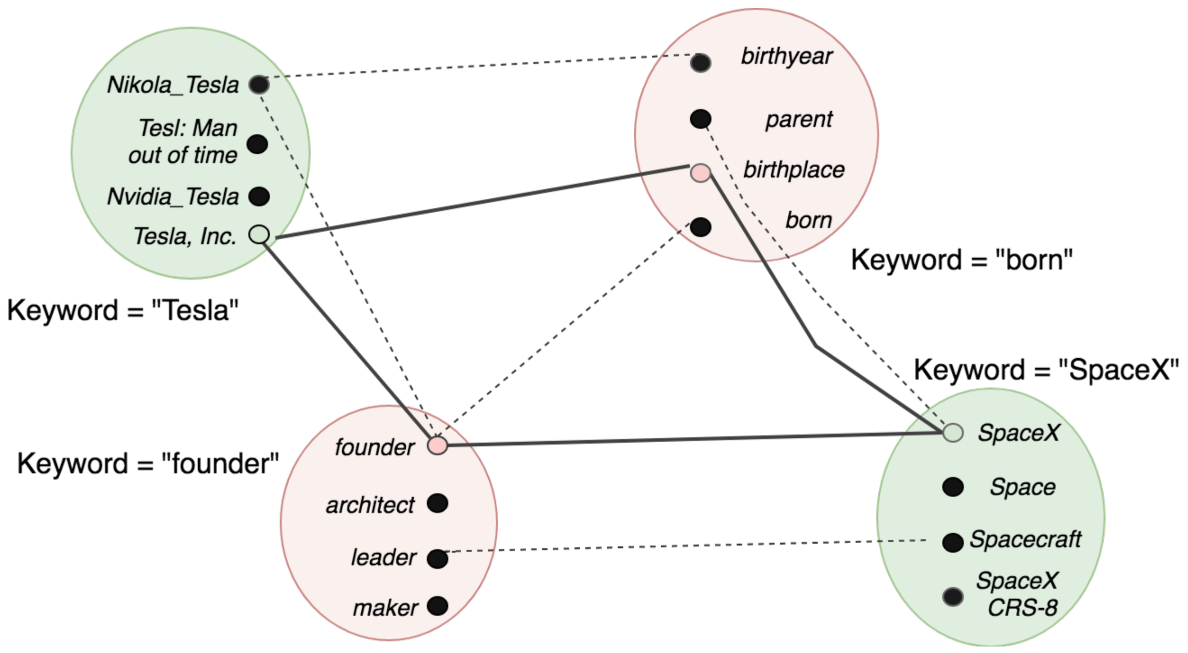


Figure 7.3: Using GTSP for disambiguation : The bold line represents the solution offered by the GTSP solver. Each edge represents an existing connection in the knowledge graph. The edge weight is equal to the number of hops between the two nodes in the knowledge graph. We also add the index search ranks of the two nodes the edges connect to the edge weight when solving for GTSP.

While a GTSP based solution would be suitable for solving the joint entity and relation linking problem, it has the drawback that it can only provide the best candidate for each keyword given the list of candidates. Most approximate GTSP solutions do not explore all possible paths and nodes and hence a comprehensive scoring and re-ranking of nodes is not possible. Ideally, we would like to go beyond this and re-rank all candidates for a given keyword. This would open up new opportunities from a QA perspective, i.e. a user could be presented with a sorted list of multiple possible answers to select from.

7.2.3 Using Connection Density for Disambiguation

As discussed earlier, once the candidate list generation is achieved, EARL offers two independent modules for the entity and relation linking. In the previous subsection, we discussed one approach using GTSP. In this subsection we will discuss the second approach for disambiguation using Connection Density, which works as an alternative to the GTSP approach. We have also compared the two methods in Table 7.2.

Formalization of Connection Density

For identified keywords in a question we have the set $\mathcal{K}$ as defined earlier. For each keyword K_i we have list L_i which consists of all the candidate uris generated by text search. We have n such candidate lists for each question given by, $\mathcal{L} = \{L_1, L_2, L_3, ..., L_n\}$. We consider a probable candidate $c_m^i \in L_i$, where m is the total number of candidates to be considered per keyword, which is the same as the number of items in each list.

The hop distance $dKGhops(c_i^k, c_j^o) \in \mathbb{Z}^+$ is number of hops between c_i^k and c_j^o in the subdivision knowledge graph. If the shortest path from c_i^k and c_j^o requires the traversal of h edges then $dKGhops(c_i^k, c_j^o) =$

GTSP	Connection Density
Requires no training data	Requires data to train the XGBoost classifier
The approximate GSTP LKH solution is only able to return the top result as not all possible paths are explored.	Returns a list of all possible candidates in order of score
Time complexity of LKH is $O(nL^2)$ where n = number of nodes in graph, L = number of clusters in graph of	Time complexity is $O(N^2L^2)$ where N = number of nodes per cluster, L = number of clusters in graph
Relies on identifying the path with minimum cost	Depends on identifying dense and short-hop connections

Table 7.2: Comparison of GTSP based approach and Connection density for Disambiguation

h.

Connection Density is based on the three features: Text similarity based initial Rank of the List item ($\mathcal{R}_i$) Connection-Count (C) and Hop-Count ($\mathcal{H}$)

Initial Rank of the List ($\mathcal{R}_i$), is generated by retrieving the candidates from the search index via text search. This is achieved in the preprocessing steps as mentioned in the earlier sections. Further, to define C we introduce *dConnect*.

$$dConnect(c_i^k, c_j^o) = \begin{cases} 1 & \text{if } dKGhops(c_i^k, c_j^o) \leqslant 2 \\ 0 & \text{otherwise} \end{cases} \tag{7.2}$$

The Connection-Count C for an candidate c, is the number of connections from c to candidates in all the other lists divided by the total number n of keywords spotted. We consider nodes at hop counts of greater than 2 disconnected because nodes too far away from each other in the knowledge base do not

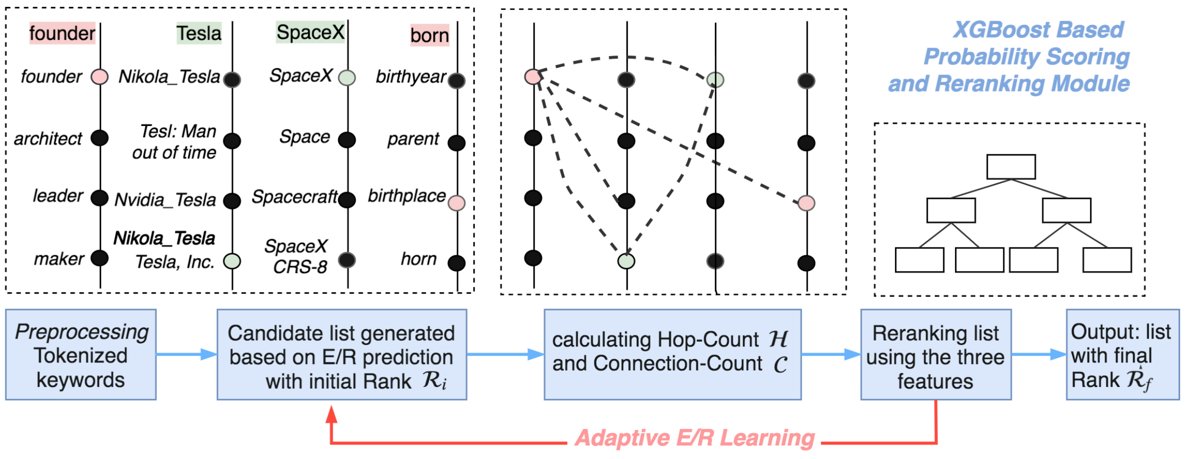

founder
Tesla
SpaceX
born
founder
Nikola_Tesla
SpaceX
birthyear
architect
Tesl: Man out of time
Space
parent
leader
Nvidia_Tesla
Spacecraft
birthplace
maker
Nikola_Tesla
Tesla, Inc.
SpaceX CRS-8
horn
XGBoost Based Probability Scoring and Reranking Module
Preprocessing Tokenized keywords
Candidate list generated based on E/R prediction with initial Rank $\mathcal{R}_i$
calculating Hop-Count $\mathcal{H}$ and Connection-Count C
Reranking list using the three features
Output: list with final Rank $\mathcal{R}_f$
Adaptive E/R Learning

Figure 7.4: Connection Density with example: The dotted lines represent corresponding connections between the nodes in the knowledge base.

carry meaningful semantic connection to each other.

$$C(c_i^k) = 1/n \sum_{o|o\neq k} \sum_{j=1}^{j=m} dConnect(c_i^k, c_j^o) \tag{7.3}$$

The Hop-Count $\mathcal{H}$ for a candidate c, is the sum of distances from c to all the other candidates in all the other lists divided by the total number of keywords spotted.

$$\mathcal{H}(c_i^k) = 1/n \sum_{o|o\neq k} \sum_{j=1}^{j=m} dKGhops(c_i^k, c_j^o) \tag{7.4}$$

Candidate Re-ranking

$\mathcal{H}, C$ and $\mathcal{R}_i$ constitute our feature space $\mathcal{X}$. This feature space is used to find the most relevant candidate given a set of candidates for an identified keyword in the question. We use a machine learning classifier to learn the probability of being the most suitable candidate $\bar{\mathbf{c}}^i$ given the set of candidates. The final list $\mathcal{R}_f$ is obtained by re-ranking the candidate lists based on the probability assigned by the classifier. Ideally, $\bar{\mathbf{c}}^i$ should be the top-most candidate in $\mathcal{R}_f$.

The training data consists of the features $\mathcal{H}, C$ and $\mathcal{R}_i$ and a label 1 if the candidate is the correct, 0 otherwise. For the testing, we apply the learned function from the classifier f on $\mathcal{X}$ for every candidate $\in c_i$ and get a probability score for being the most suitable candidate. We perform experiments with three different classifiers, namely extreme gradient boosting(xgboost), SVM(with a linear kernel) and logistic regression to re-rank the candidates. Figure 7.5 reports the performance of the classifiers for MRR. on top-10 fetched candidate lists. The experiments are done using a 5-fold cross-validation strategy where, for each fold we train the classifier on the training set and observe the mean reciprocal rank (MRR) of $\bar{\mathbf{c}}^i$ on the testing set after re-ranking the candidate lists based on the assigned probability. The average MRR on 5-fold cross-validation for the three classifiers are 0.905, 0.704 and 0.794 respectively. Hence, we use xgboost as the final classifier in our subsequent experiments for re-ranking.

Algorithm

We now present a pseudo-code version of the algorithm to calculate the two features: Connection Density algorithm is used for finding hop count and connection count for each candidate node. We then pass these features to a classifier for scoring and ranking This algorithm (Algorithm 1 Connection Density) has a time complexity given by $O(N^2L^2)$ where N is the number of keywords and L is the number of candidates for each keyword.

7.2.4 Adaptive E/R Learning

EARL uses a series of sequential modules with little to no feedback across them. Hence, the errors in one module propagate down the line. To trammel this, we implement an adaptive approach especially for curbing the errors made in the pre-processing modules. While conducting experiments, it was observed that most of the errors are in the shallow parsing phase, mainly because of grammatical errors in LC-QuAD which directly affects the consecutive E/R prediction and candidate selection steps. If the E/R prediction is erroneous, it will search in a wrong Elasticsearch index for probable candidate list generation. In such a case none of the candidates $\in c^i$ for a keyword would contain $\bar{\mathbf{c}}^i$ as is reflected by the probabilities assigned to c^i by the re-ranker module. If the maximum probability assigned to c^i is less

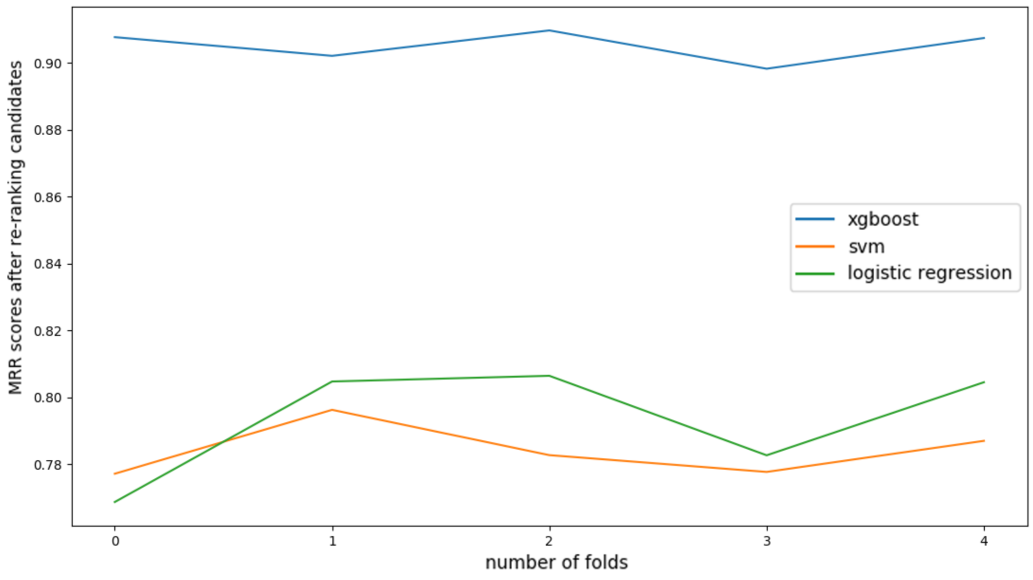


Figure 7.5: MRR. with different algorithms on cross-validation sets

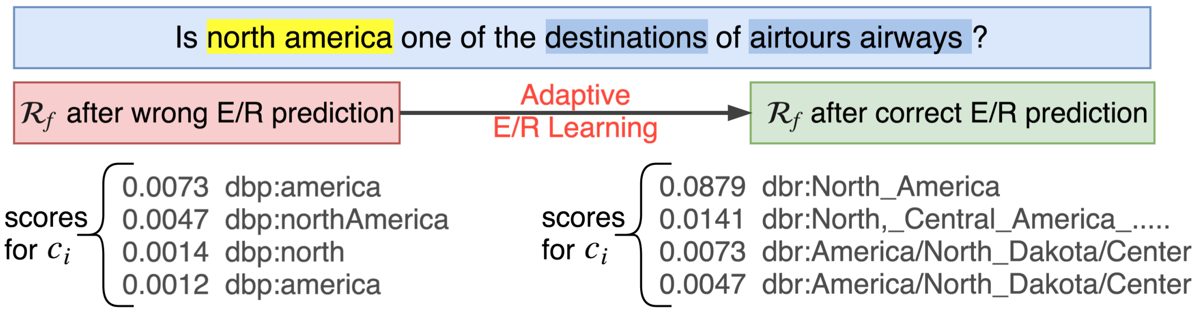

Is north america one of the destinations of airtours airways ?
$\mathcal{R}_f$ after wrong E/R prediction
Adaptive E/R Learning
$\mathcal{R}_f$ after correct E/R prediction
scores for c_i
0.0073 dbp:america
0.0047 dbp:northAmerica
0.0014 dbp:north
0.0012 dbp:america
scores for c_i
0.0879 dbr:North_America
0.0141 dbr:North,_Central_America_.....
0.0073 dbr:America/North_Dakota/Center
0.0047 dbr:America/North_Dakota/Center

Figure 7.6: Adaptive E/R learning

than a very small threshold value, empirically chosen as 0.01, we re-do the steps from ER prediction after altering the original prediction. If the initial assigned probability is *entity*, we change it to *relation* and vice-versa, example Figure 7.6. This module is empirically evaluated in Table 7.5.

7.3 Evaluation

Data Set

: LC-QuAD [16] is the largest complex questions data set available for QA over KGs. We have annotated this data set to create a gold label data set for entity and relation linking, i.e. each question now contains the correct KG entity and relation URIs with their respective text spans in the question. This annotation was done in a semi-automated process and subsequently manually verified. The annotated dataset of 5000 questions is publicly available at `https://figshare.com/projects/EARL/28218`

Algorithm 2 Connection Density

```
function : ConnectionDensity( )
input    : $\mathcal{L}$ , with $n$ number of keywords // an array of arrays
output   : Hop-Count $\mathcal{H}$, Connection-Count $C$
dConnectCounter = { } // Count for connections from and to each node
dHopCounter = { } // Similarly hop counts for each node
foreach $L_a \in \mathcal{L}$ do
    foreach $c_i^a \in L_a$ do
        $dConnectCounter[c_i^a] = 0$ // Initializing the dictionary
        $dHopCounter[c_i^a] = 0$
    end
end
foreach $(L_a, L_b) \in \mathcal{L}$ do
    foreach $c_i^a \in L_a$ do
        foreach $c_j^b \in L_b$ do
            if $dKGhops(c_i^a,c_j^b) <= 2$ then
                $dConnectCounter[c_i^a]$ += 1
                $dConnectCounter[c_j^b]$ += 1
            end
            $dHopCounter[c_i^a]$ += dKGhops($c_i^a$,$c_j^b$)
            $dHopCounter[c_j^b]$ += dKGhops($c_i^a$,$c_j^b$)
        end
    end
end
foreach $(c_i, score) \in dConnectCounter$ do
    $C(c_i) = dConnectCounter(c_i)/n$ // Normalization with respect to number of
        keywords spotted
end
foreach $(c_i, score) \in dHopCounter$ do
    $\mathcal{H}(c_i) = dHopCounter(c_i)/n$
end
return (Hop-Count $\mathcal{H}$, Connection-Count $C$)
```

7.3.1 Experiment 1: Comparison of GTSP, LKH and Connection Density

Aim: We evaluate hypotheses (**H1** and **H2**) that the connection density and GTSP can be used for joint linking task. We also evaluate the LKH approximation solution of GTSP for doing this task. We compare the time complexity of the three different approaches.

Results: Connection density results in a similar accuracy as that of an exact GTSP solution with a better time complexity (see Table 7.7). Connection density has worse time complexity than approximate GTSP solver LKH if we assume the best case of equal cluster sizes for LKH. However, it provides a better accuracy. Moreover, the average time taken in EARL using connection density (including the candidate generation step) is 0.42 seconds per question.

Further observing Table 7.7, we can see that the brute force GTSP solution and Connection Density have similar accuracy, but the brute force GTSP solution has exponential time complexity. The approximate solution LKH has polynomial run time, but its accuracy drops compared to the brute force GTSP solution. Moreover, from a question answering perspective the ranked list offered by the Connection Density approach is useful since it can be presented to the user as a list of possible correct solutions or used by subsequent processing steps of a QA system. Hence, for further experiments in this section we used the connection density approach.

7.3.2 Experiment 2: Evaluating Joint Connectivity and Re-ranker

Aim: Evaluating the performance of Connection Density for predicting the correct entity and relation candidates from a set of possible E-R candidates. Here we evaluate hypothesis **H2**, the correct candidates exhibit relatively dense and short-hop connections.

Metrics: We use the mean reciprocal rank of the correct candidate $\bar{\mathbf{c}}^i$ for each entity/relation in the query. From the probable candidate list generation step, we fetch a list of top candidates for each identified phrase in a query with a k value of 10, 30, 50 and 100, where k is the number of results from text search for each keyword spotted. To evaluate the robustness of our classifier and features we perform two tests. i) On the top half of Table 7.4 we re-rank the top k candidates returned from the previous step. ii) On the bottom half of Table 7.4 we artificially insert the correct candidate into each list to purely test re-ranking abilities of our system (this portion of the table contains k^* as the number of items in each candidate list). We inject the correct uris at the lowest rank (see k^*), if it was not retrieved in the top k results from previous step.
Results: The results in Table 7.4 depict that our algorithm is able to successfully re-rank the correct

Approach	**Accuracy (K=30)**	**Accuracy (K=10)**	**Time Complexity**
Brute Force GTSP	0.61	0.62	$O(n^2 2^n)$
LKH - GTSP	0.59	0.58	$O(nL^2)$
Connection Density	0.61	0.62	$O(N^2 L^2)$

Table 7.3: Empirical comparison of Connection Density and GTSP: n = number of nodes in graph; L = number of clusters in graph; N = number of nodes per cluster; top K results retrieved from ElasticSearch.

Value of k	$\mathcal{R}_f$ **based on** $\mathcal{R}_i$	$\mathcal{R}_f$ **based on** $\mathcal{C}, \mathcal{H}$	$\mathcal{R}_f$ **based on** $\mathcal{R}_i, \mathcal{C}, \mathcal{H}$
$k = 10$	0.543	0.689	0.708
$k = 30$	0.544	0.666	0.735
$k = 50$	0.543	0.617	**0.737**
$k = 100$	0.540	0.534	0.733
$k^* = 10$	0.568	0.864	**0.905**
$k^* = 30$	0.554	0.779	0.864
$k^* = 50$	0.549	0.708	0.852
$k^* = 50$	0.545	0.603	0.817

Table 7.4: Evaluation of joint linking performance

System	Accuracy LC-QuAD	Accuracy - QALD
FOX [150] + AGDISTIS [19]	0.36	0.30
DBpediaSpotlight [18]	0.40	0.42
TextRazor[6]	0.52	0.53
Babelfy [87]	0.56	0.56
EARL without adaptive learning	0.61	0.55
EARL with adaptive learning	**0.65**	**0.57**

Table 7.5: Evaluating EARL's Entity Linking performance

System	Accuracy LC-QuAD	Accuracy QALD
ReMatch [102]	0.12	0.31
RelMatch [151]	0.15	0.29
EARL without adaptive learning	0.32	0.45
EARL with adaptive learning	**0.36**	**0.47**

Table 7.6: Evaluating EARL's Relation Linking performance

URIs if the correct ones are already present. In case correct URIs were missing in the candidate list, we inserted URIs artificially as the last candidate . The MRR then increased from 0.568 to 0.905.

7.3.3 Experiment 3: Evaluating Entity Linking

Aim: To evaluate the performance of EARL with other state-of-the-art systems on the entity linking task. This also evaluates our hypothesis **H3**.
Metrics: We are reporting the performance on accuracy. Accuracy is defined by the ratio of the correctly identified entities over the total number of entities present.
Result: EARL performs better entity linking than the other systems (Table 7.5), namely Babelfy, DBpediaSpotlight, TextRazor and AGDISTIS + FOX (limited to entity types - LOC, PER, ORG). We conducted this test on the LC-QuAD and QALD-7 dataset[5]. The value of **k** is set to 30 while re-ranking and fetching the most probable entity.

7.3.4 Experiment 4: Evaluating Relation Linking

Aim: Given a question, the task is to the perform relation linking in the question. This also evaluates our hypothesis **H3**.
Metrics: We use the same accuracy metric as in the Experiment 3
Results: As reported in Table 7.6, EARL outperforms other approaches we could run on LC-QuAD and QALD. The large difference in accuracy of relation-linking over LC-QuAD over QALD, is due to the face that LC-QuAD has 82% questions with more than one relation, thus detecting relation phrases in the question was difficult.

[5]https://project-hobbit.eu/challenges/qald2017/

7.3.5 Experiment 5: Blooms for Response Time

Aim: We empirically show the query-time speed up achieved by the usage of Bloom filter compared to SPARQL over KG in terms of response time. In our experiment we checked the connectivity between two nodes of the knowledge graph with a fixed path length. Say we check connectivity entity e_1 and e_2 with a fixed length of 2 hops (e_1 to x and x to e_2, where x is any predicate between two entities).
Results: We observe that the SPARQL response time (3300 micro seconds) for such a query is many folds higher then the Bloom filter's response time (17 micro seconds). The results in Table 7.7 show that the response time of SPARQL increases with the hop length, where as the Bloom filter results are constant. However it must be added that the one-time construction of the Bloom filters as a pre-processing step requires several hours to complete.

Connection check with path-length	SPARQL infer time	Bloom filter infer time
1-hop e_1 - p_1	3300 μsec	17 μsec
2-hop e_1 - $?x$ - e_2	3300 μsec	17 μsec
3-hop e_1 - $?x$ - $?y$ - p_2	8500 μsec	17 μsec
4-hop e_1 - $?x$ - $?y$ - $?z$ - e_3	11059 μsec	17 μsec

Table 7.7: Bloom Filter's Inference Time Compared to SPARQL

7.4 Availability

EARL API is publicly available at `https://earldemo.sda.tech`. The EARL API takes the natural language question as input, and the returns a ranked list of the URIs for the corresponding keyword phrases. A demo of EARL is also accessible via EARL's homepage mentioned above. The ranked list output by EARL also consists of *confidence scores*. EARL's connection density module has a time complexity of $O(N^2L^2)$, where N is the number of keyword phrases, and L is the size of list retrieved for candidate keyword phrase. The number of bloom queries made for a question is given by the equation $(L^2(N^2 - N)/2)$. EARL API has an average response time of 0.42 seconds per question when queried on the same machine hosting the service. The source code for EARL is available at `https://github.com/AskNowQA/EARL`. Along with the code, we also release the extended label vocabulary.

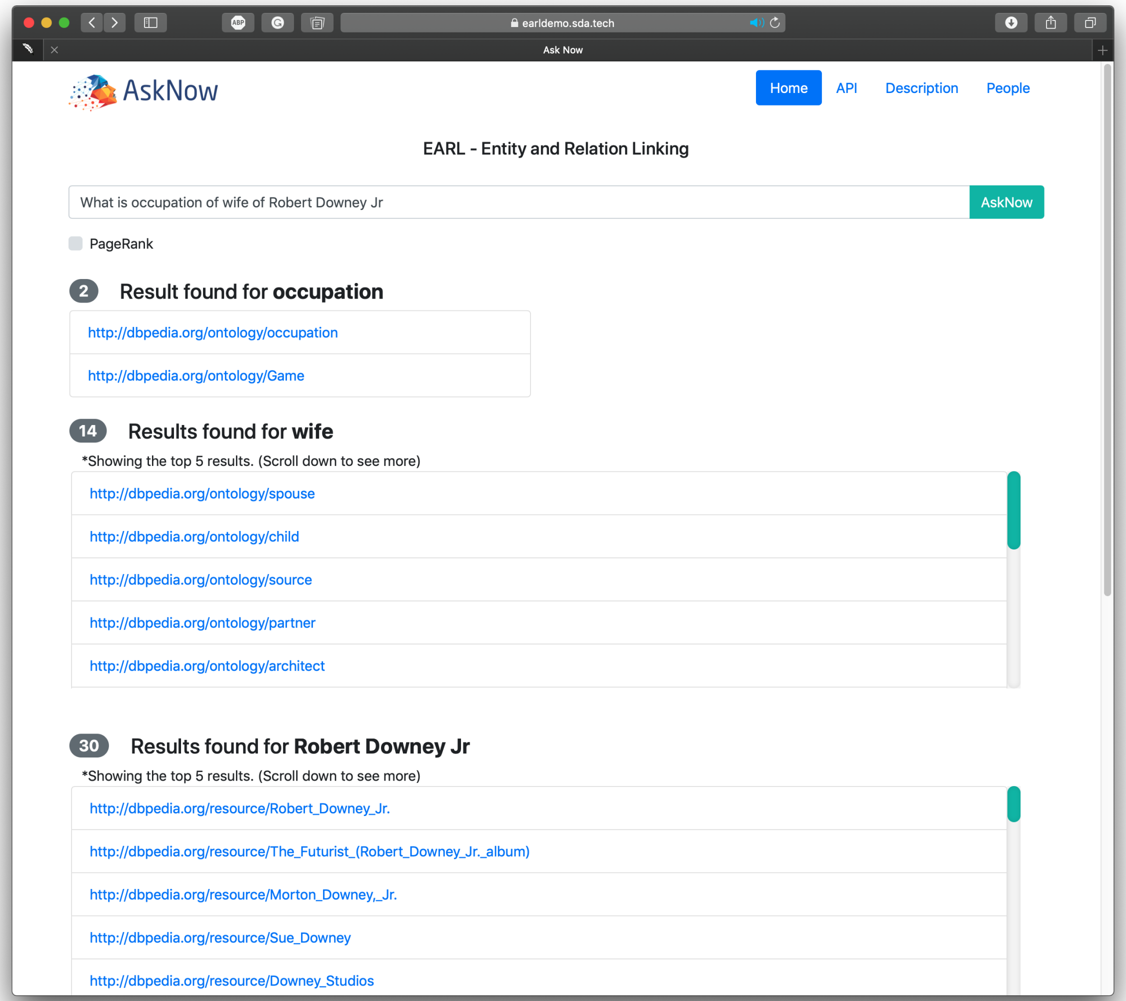
7.5 Discussion

Our analysis shows that we have provided two tractable (polynomial with respect to the number of clusters and the elements per cluster) approaches of solving the joint entity and relation linking problem. We experimentally achieve similar accuracy as the exact GTSP solution with both LKH-GTSP and Connection Density with better time complexity, which allows us to use the system in QA engines in practice. It must be noted that one of the salient features of LKH-GTSP is that it requires no training data for the disambiguation module while on the other hand Connection Density performs better given training data for its XGBoost classifier. While the system was tested on DBpedia, it is not restricted to a particular knowledge graph.

There are some limitations: The current approach does not tackle questions with hidden relations, such as "How many shows does HBO have?". Here the semantic understanding of the corresponding SPARQL query is to count all TV shows (*dbo:TelevisionShow*) which are owned by (*dbo:company*) the HBO (*dbr:HBO*). Here *dbo:company* is the hidden relation which we do not attempt to link. However, it could be argued that this problem goes beyond the scope of relation linking and could be better handled by the query generation phase of a semantic QA system.

Another limitation is that EARL cannot be used as inference tool for entities as required by some questions. For example Taikonaut is an astronaut with Chinese nationality. The system can only link taikonaut to *dbr:Astronaut*, but additional information can not be captured. It should be noted, however, that EARL can tackle the problem of the "lexical gap" to a great extent as it uses synonyms via the grammar inflection forms.

Our approaches of LKH-GTSP and Connection Density both have polynomial and approximately similar time complexities. EARL with either Connection Density or LKH-GTSP can process a question



EARL - Entity and Relation Linking

What is occupation of wife of Robert Downey Jr

PageRank

2 Result found for **occupation**

http://dbpedia.org/ontology/occupation

http://dbpedia.org/ontology/Game

14 Results found for **wife**

*Showing the top 5 results. (Scroll down to see more)

http://dbpedia.org/ontology/spouse

http://dbpedia.org/ontology/child

http://dbpedia.org/ontology/source

http://dbpedia.org/ontology/partner

http://dbpedia.org/ontology/architect

30 Results found for **Robert Downey Jr**

*Showing the top 5 results. (Scroll down to see more)

http://dbpedia.org/resource/Robert_Downey_Jr.

http://dbpedia.org/resource/The_Futurist_(Robert_Downey_Jr._album)

http://dbpedia.org/resource/Morton_Downey,_Jr.

http://dbpedia.org/resource/Sue_Downey

http://dbpedia.org/resource/Downey_Studios

Figure 7.7: Showcasing the EARL Demo Webpage

in a few hundred milliseconds on a standard desktop computer on average. The result logs, experimental setup and source code of our system are publicly available at: https://github.com/AskNowQA/EARL.

7.6 Conclusions

In this chapter, we developed a KGQA module. Here we propose EARL, a framework for joint entity and relation linking. We treat the entity and relation linking as a classical three-step process which comprises of span detection, candidate generation and disambiguation. EARL is based on DBpedia and LC-QuAD 1.0 dataset but can be easily ported to other KGs too. For entity and relation candidate generation, we use string similarity; thus, we created huge dictionaries of labels to cover all syntactic and semantic variations. Labels of entities are extended by adding labels from Wikidata. For extension of relation labels, we go further and use synonyms and word embeddings. For the third step of disambiguation, we provide two strategies for joint linking - one based on reducing the problem to an instance of the Generalized Travelling Salesman Problem and the other based on a connection density-based machine learning approach. We also introduced a feedback mechanism in the later stage of the EARL pipeline to correct the errors of the early stages. Our experiments on QA benchmarks resulted in accuracies which are significantly above the results of current state-of-the-art approaches for entity and relation linking. Future work could be in the candidate generation phase to ensure that a higher proportion of correct candidates are retrieved. Another way to improve EARL could be to move the system towards more end-to-end architecture.

CHAPTER 8

Question Aware Answer Verbalization

In Chapter 4 we developed a KGQA system with rule based approach. In Chapter 5 and 6 we generated large scale datasets for KGQA, to support the research community and provide the resource for advancing the State of the art. Chapter 7 uses that dataset in entity linking and relation linking for question over KG with state of the art results. The overall objective of the KGQA is to get a concise answer for a natural language user question. The above mentioned datasets and QA system retrieve answer from the KG as a node from the KG node(entity or a list of entity) or as literal(Boolean, numbers, timestamp, string or a list them). These answers are concise and to the point, but they lack correct answer representation for a human like response, which is a goal in Artificial Intelligence.

In this chapter we shed light on answer representation aspect of question answering. Our objective here is to move away from template based NLG approaches for answer verbalization and rather advance with machine learning based solution.

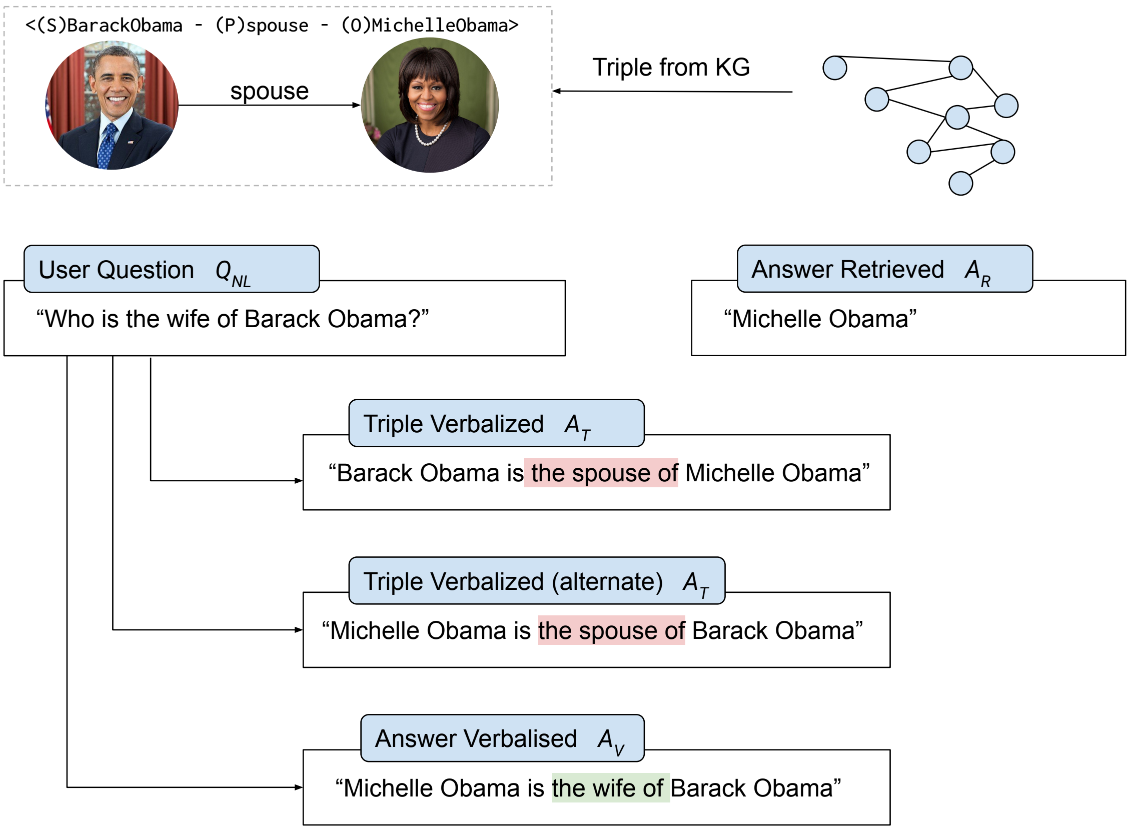
Research Question 5 (RQ5)

How to improve the answer representation by using the context and textual description from the user question?

The following are key contributions of this chapter:

- Provision of a large-scale dataset of over 100k questions along with their corresponding verbalized answers.
- We present a cost-efficient semi-automatic framework for generating large-scale answer verbalization datasets, which was also used to generate VANiLLa.
- We provide multiple baseline models based on the conventional sequence-to-sequence framework as reference for future research.
- We also provide empirical proof that answer verbalizations are better suited response and provide more context compared to RDF triple verbalizations.

Question Answering over Knowledge Graphs (KGQA) performs the task to automatically answer natural language questions posed by users. They accomplish this by first understanding the question in natural language and then retrieving answers from a Knowledge Graph (KG) to generate concise



<(S)BarackObama - (P)spouse - (O)MichelleObama>
spouse
Triple from KG
User Question Q_{NL}
"Who is the wife of Barack Obama?"
Answer Retrieved A_R
"Michelle Obama"
Triple Verbalized A_T
"Barack Obama is the spouse of Michelle Obama"
Triple Verbalized (alternate) A_T
"Michelle Obama is the spouse of Barack Obama"
Answer Verbalised A_V
"Michelle Obama is the wife of Barack Obama"

Figure 8.1: Example showing answer verbalization using context from question and comparing it to RDF verbalization.

responses. KGQA systems are also being used as part of Dialogue Systems where users ask questions to the system during their conversation and the system aims to answer these using the KG. However, the generated responses from these systems are often just provided in the form of resources or literals rather than full sentences. To overcome this inadequacy, some systems adapt a template-based approach for producing verbalized answers. The template based approaches are limited to the rules or templates built by field experts requiring manual expertise.

To overcome the shortcomings of the template based approach, machine learning techniques for verbalizations are necessary. These techniques, in turn, depend primarily on large-scale training data. Consequently, datasets play a vital role in the evolution of ML techniques. At present, the datasets that can be applied for answer verbalization tasks are mostly RDF verbalization datasets like WebNLG [152]. The WebNLG dataset aims to transform RDF triples from DBpedia [153] into natural language sentences.

The most relevant RDF triples matching the user question is retrieved by the QA system, which returns a resource or literal as a response. In this regard, an RDF verbalization model, converting RDF triples in the form of `<subject-predicate-object> (S,P,O)` into natural language sentences, can be integrated to the QA pipeline for generating the response in natural language. In Figure 8.1 we show a potential response of a QA system when asked the question Q_{NL}: *"Who is the wife of Barack Obama?"*. In this example, the system fetches the RDF triple `<(S)BarackObama-(P)spouse-(O)MichelleObama>`. In terms of the user question Q_{NL}, `<Barack Obama>` is the entity present in the question, `<spouse>` is the question relation and `<Michelle Obama>` is the retrieved answer A_R. To present a verbalized response one approach would be by verbalizing this triple, the most common generation would be A_T: *"Barack Obama is the spouse of Michelle Obama"* or alternatively, *"Michelle Obama is the spouse of Barack Obama"*. In both these cases, the RDF triple verbalizations are limited to the vocabulary of the KG labels, as in this case the response consists of "spouse" whereas the question contains "wife". Hence, these verbalizations are far from what a human would respond. From this standpoint, a straightforward triple verbalization may not produce satisfactory answer verbalizations. To overcome these problems, our solution strategy is to verbalize the answer keeping in mind the context of the question by using parts of the input question. Thus, a more human-like response, shown as A_V in the figure, could be generated: *"Michelle Obama is the wife of Barack Obama"*. This generation also enable users with more informative answers, since the answer uses similar terms in the vocalizations to those in the input question, and the system response is more human-like. In order to spur research into such context-driven answer vocalizations, we present the VANiLLa: 'Verbalized Answers in Natural Language at Large scale' where the verbalized answer sentences are generated by using the context of question to encourage human-like responses.

This Chapter is organized into the following sections: (2) Problem Statement, (3) Relevance of our dataset and its possible impact along with some Related Work, (4) Dataset Generation Workflow, (5) Characteristics of the Dataset, (6) Baseline Models, (7) Evaluation results, and (8) Conclusion and Future Work.

8.1 Problem Statement

Formally, the KGQA solutions are based on semantic matching of Natural Language Question Q_{NL} to KG triples. The goal here is to generate Formal expression Q_{FL}, such that Q_{NL} and Q_{FL} have an equivalent semantic interpretation. Results retrieved through a formal query (often SPARQL in KGQA) are often either Entities, Literal or Boolean (which is a special case of Literal that we treat separately) or a list of them. Giving these as response directly to a user question is sufficient for the answer but verbalized answer will be more human friendly, which is a core objective of a KGQA system. Thus, the need for verbalizing answers in natural language in two fold: (i) to provide some context of the question in the answer to enable users to verify the provided answer [154], and (ii) to mimic human conversations. From this perspective, a new model, to generate the final representation of answer, is imperative in the QA pipeline as shown in Figure 8.2.

Formalization: KGQA Answers

Given a language question Q_{NL} KGQA Answers aims to retrieve a natural language answer A_{NL} to model a transformation function $f_{kgqa\_answers}$ such that :

$$f_{kgqa\_answers} : Q_{NL} \mapsto A_{NL}$$

$$(Q_{NL})^{C_q} \equiv (A_{NL})^{C_a}$$

C_q is the linguistic and semantic context of the natural language question
C_a is the linguistic and semantic context of the natural language answer

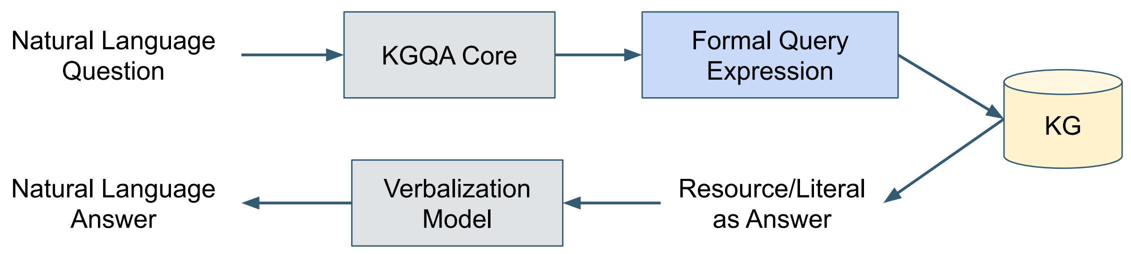


Figure 8.2: The QA pipeline for generating the answer in natural language

Within the scope of this Chapter, we do not retrieve answer from a KG for a given question. Rather we focus on verbalizing the answers retrieved through the Q_{FL} corresponding to the Q_{NL}. We describe this model as follows:

formalization: KGQA Answers

Given a user question Q_{NL} and an answer retrieved A_R by a QA system using Q_{FL}. The task of the model f is to generate verbalized answer A_V such that the context of the question is intact and we have more human-like response. Formally,

$$A_V = f(Q_{NL}, A_R)$$

We further introduce our hypothesis regarding the semantic closeness of natural language question(input to KGQA system) and natural language answer(output of KGQA system).

Hypothesis H1

: For a given question Q_{NL} and the retrieved answer A_R from KG triples T, the verbalized answer A_V with question context is semantically and syntactically closer to the question Q_{NL} in comparison with that of the KG triple verbalization A_T and Q_{NL}. Formally,

$$Sim(A_V, Q_{NL}) > Sim(A_T, Q_{NL}) \tag{8.1}$$

Similarly, for a given question Q_{NL} and the retrieved answer A_R from KG triples T, the RDF verbalization A_T is semantically and syntactically closer to the triple T in comparison with that of the answer verbalization A_V and T. Formally,

$$Sim(A_T, T) > Sim(A_V, T) \tag{8.2}$$

where, Sim is the similarity measure.
We empirically prove this hypothesis with Experiment 1 in Section: 8.6.1.

8.2 Relevance and Related Datasets

To understand the impact of our dataset we first showcase the previous works that have been done in this domain. We report some of the resources which are currently available for verbalizing answers. GENQA [114] and COREQA [113] are two datasets in Chinese language which provide answers in the form of sentences rather than a concise answer. To the best of our knowledge, VQuAnDa [115] is the first and only dataset in English language to provide answer verbalization. The dataset contains questions from LC-QuAD [155] on DBpedia as the target KG. The advantage of this dataset is doubled due to the availability of SPARQL representation for questions. Thus, it is also useful for model's verbalizing answers from a SPARQL query. The dataset was aimed to enable users to validate the information generated from QA systems by incorporating not only the retrieved information but also additional characteristics indicating the process of answer retrieval. Nevertheless, this dataset does not cover key features of the user question to mimic human conversation in addition to enable answer validation.

VANiLLa dataset is a large scale dataset with over 100k question answer pair compared to the 5k in VQuAnDa dataset. The VQuAnDa aims to generates the position of the answer in the natural language sentence hence in return has to implement a slot filling approach to achieve the final response. Whereas, VANiLLa promotes a end-to-end approach by avoid any post processing of the generated response. Another aspect of the VANiLLa is that it is KG independent unlike VQuAnDa whose target KG is DBpedia. These features of our dataset clearly indicates it's relevance in the NLP community. In this regard, our dataset would impact the following sectors of the NLP community:

- **Natural Language Generation**: Popular NLG tasks and dataset on RDF verbalzation, such as WebNLG[152], do not take into consideration the question elements when reconstructing the facts in natural language. Hence, the sentences generated involve relevant information from the facts without any context of the question. From this standpoint, our dataset would open up new paradigm in NLG research.

- **Question Answering and Dialog Systems**: Different QA datasets based on Text, such as SQuAD [156] and NaturalQuestions [157], and KG, such as SimpleQuestion [82], WebQuestionSP [158], ComplexWebQuestions [83], QALD-9 [13], LC-QuAD 1.0 [155], LC-QuAD 2.0 [17], exists. Furthermore, datasets are also adapted into conversation or dialog format with multiple utterances such as, QuAC [159], CoQA [160], CSQA [161] and SQA [162]. Recent trend in QA datasets are based on the following aspects: (1) expanding the size of dataset, (2) extending the dataset with more variations in question type such as aggregations, boolean queries, etc., (3) enhancing question complexity using compound features like comparison or unions, (4) enhancements covering multiple utterances in the form of conversations.

At present, the datasets associated with the task of QA or dialog systems provide concise or to-the-point answers for questions posed by users. However, verbalization of answers, which enable a more human-like response by incorporating additional information, are neglected in these datasets. From this viewpoint, VANiLLa will provide a more contextual and human-like response to user questions, using elements from the question to generate the verbalized answers. Furthermore, it is beneficial in dialog systems to have a continuous flow in the conversation, by enabling users with more content for the next utterance.

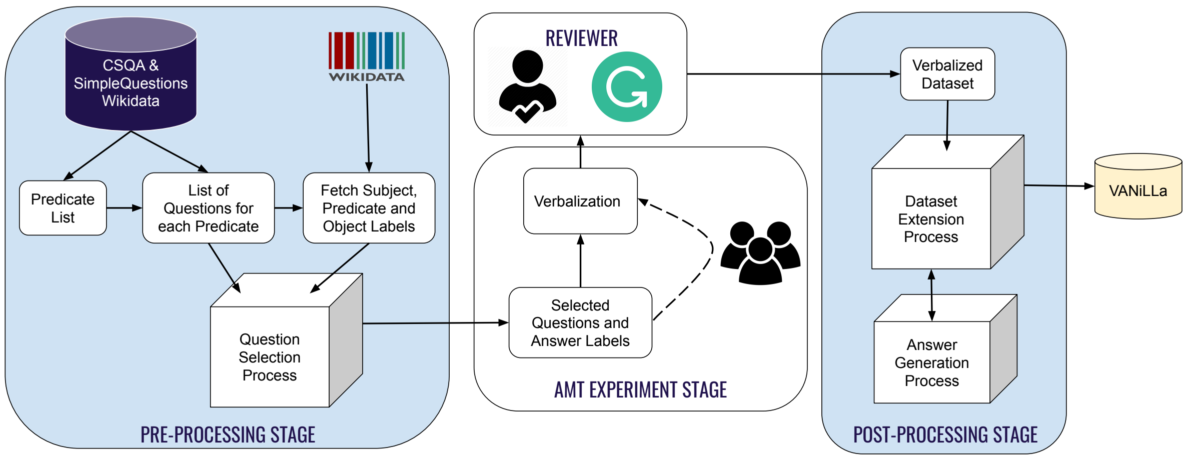

CSQA & SimpleQuestions Wikidata
WIKIDATA
Predicate List
List of Questions for each Predicate
Fetch Subject, Predicate and Object Labels
Question Selection Process
PRE-PROCESSING STAGE
REVIEWER
Verbalization
Selected Questions and Answer Labels
AMT EXPERIMENT STAGE
Verbalized Dataset
Dataset Extension Process
Answer Generation Process
POST-PROCESSING STAGE
VANiLLa

Figure 8.3: Dataset Generation Workflow

8.3 VANiLLa Dataset Generation Workflow

Our dataset, **VANiLLa** (**Verbalized Answers in Natural Language at Large scale**) consists of simple questions Q_{NL} and answers A_R along with their answers in natural language A_V. The simple questions were selected from the CSQA [161] and SimpleQuestionsWikidata [163] datasets. These datasets are Question Answering over Knowledge Graph (KGQA) datasets on Wikidata [164]. The verbalized answers were generated using Amazon Mechanical Turk (AMT). The dataset creation process workflow, as described in Figure 8.3, consists of four stages: (i) Pre-Processing Stage, (ii) AMT Experiment Stage, (iii) Reviewer, and (iv) Post-Processing Stage.

8.3.1 Preprocessing Stage

In this stage, we first collected all simple questions from CSQA[1] and SimpleQuestionsWikidata datasets. While filtering questions, we ignore questions which produce more than one entity as an answer. After the filtering process, we generate a list of predicates present in the question for each dataset. On the basis of these predicates, we group the questions for a particular predicate to form a list. We then fetch labels for entities and relations present in the question and the answer from Wikidata. These labels along with the question lists are sent to the question selected process. The purpose of this pre-processing step is two fold: (i) reducing the cost of crowd-sourcing the dataset from scratch, and (ii) opening the possibility of extending the CSQA and SimpleQuestionsWikidata or SimpleQuestions dataset with the verbalized answers.

8.3.2 Question Selection Process

The process starts by picking unique questions from the list of questions for a particular relation or predicate. This process maintains a better production cost to question variation ratio. By utilizing this process, a single instance of all the similar questions are selected for the AMT experiment, thereby, the cost of verbalization reduces. Later, we may use the verbalizations of these selected questions as templates to automatically generate scale large data without the requirement manual workforce. As

[1]Note: We choose only direct questions where the KG entity directly exists in the question and no co-referencing is required.

illustrated in Algorithm 3, the process takes Q as input, which is a list of all filtered questions for a particular relation and outputs S, a list of selected questions for a particular relation. The first step is to copy the first question into the selected list S. Then we check for the 4-gram similarity between the other questions in list Q with all the selected questions in list S and continue adding dissimilar questions into list S.

Algorithm 3 VANiLLa: Question Selection Process

```
    Input: $Q = \{Q_1, Q_2, ...., Q_n\}$
    Output: $S = \{S_1, S_2, ...., S_m\}$
1:  $S \leftarrow Q_1$
2:  for all $Q_i \in Q$ do
3:      Check 4-gram similarity between $Q_i$ and $S_i$ $\forall S_i \in S$
4:      if $Q_i$ and $S_i$ are not similar $\forall S_i \in S$ then
5:          Add $Q_i$ to $S$
6:      end if
7:  end for
```

To check the similarity between two questions, we first mask the entities in both the questions and then find 4-grams for both questions. If any of the 4-grams match, then the questions are similar, otherwise they are dissimilar. For example, S_1 is *"Who is the wife of Barack Obama?"* and Q_1 is *"Who is the wife of Donald Trump?"*. The 4-grams for S_1 and Q_1 are *{"who", "is", "the", "wife"}* and *{"is", "the", "wife", "of"}*. Hence, S_1 and Q_1 are similar. Now, if we check S_1 with another question say Q_2, which is *"Who is married to Barack Obama?"*. The 4-grams for Q_2 are *{"who", "is", "married", "to"}* and do not match with S_1. Hence, they are dissimilar. Therefore, Q_2 is added into the selected list S.

8.3.3 AMT Experiment Stage

In this stage, the human workers or turkers working on AMT are provided with a question Q_{NL} from the selected list S and its answer label A_R retrieved from Wikidata [163]. The workers are instructed to rewrite the answers in natural language sentences so as to generate the answer sentence A_V without re-writing the answer label. Thus, the crowd-sourcing task aims to verbalize the answers in natural language. The turkers are provided with explicit directions to verbalize the answer using the context of the question Q_{NL}. They are also presented with adequate examples to understand the task well.

8.3.4 Reviewer

A manual reviewing step is implemented to check for grammatical errors and invalid answers verbalized in the AMT stage. One native English speaking reviewer along with the help of the grammar checking tool, Grammarly[2] made the necessary corrections.

8.3.5 Post-Processing Stage

The last stage of the generation workflow yields a cost-effective extension of the dataset by utilizing the set of verbalizations received from the previous stage. The **Dataset Extension Process** searches for all similar question for every question in the verbalized dataset using 4-gram similarity and then calls the **Answer Generation Process** which employing the verbalized answer as a template to generate the

[2]https://app.grammarly.com/

answer in natural language for all the other similar questions. Due to these actions, we can provide a large-scale dataset with minimal cost.

8.4 VANiLLa Dataset Characteristics

An example of a dataset instance is shown below:

```
{
    "question id" : 76424,
    "question" : "where did henri gouraud stop breathing",
    "answer" : "Paris",
    "answer_sentence" : "henri gourand stopped breathing in paris"
}
```

The different tags of the dataset are:

- *question-id*: an unique identification number for a dataset instance
- *question*: question Q_{NL}
- *answer*: retrieved answer A_R
- *answer_sentence*: verbalized answer in natural language A_V

Dataset	#Questions	Question type	Knowledge Graph	#Relations
VANiLLa	107166	simple	independent	329

Table 8.1: VANiLLa Dataset Characteristics

Our dataset consists of a total of 107166 simple questions, with 85732 in training set and 21434 in test set. Other characteristics of the dataset are shown in Table 8.1. One significant feature of our dataset is that it is KG-independent and hence flexible for training verbalization models of any KG. VANiLLa covers a wide range of relations (300+) providing a significant amount of variations to the dataset.

However, some drawbacks of our dataset include the non-availability of multiple answer type questions, complex questions and paraphrased questions. Also, the dataset is not in the form of conversations rather they consists of single turn questions. The dataset is still not adapted to multi-lingual question-answer pairs.

8.4.1 Reusability

Our dataset can be used in multiple research areas. The straight forward utilization is in the domain of KGQA pipeline where the final representation of the answer can be modified into a natural language sentence. This also applies for Dialog systems where the user not only expects more human-like conversation but also a scope for continuing the conversation. The dataset is KG independent, hence there is a possibility of using it for text based QA systems as well to generate answers in natural language. Also, our semi-automatic framework can be used for a cost-effective extension of the dataset. In addition, with some further investigation our dataset can be adopted in the form of multi-turns for assisting conversation type QA systems.

8.4.2 Availability and Sustainability

We have published the dataset at figshare[3] under CC BY 4.0[4] license to support sustainability. Figshare ensures data persistence and public availability, whereby guaranteeing all time accessibility of the dataset irrespective of the running status of our servers. The figshare project includes the training and the test sets our dataset.

Our baseline models and data creation framework is readily available as a open source repository[5] under a GPL 3.0[6] license. We intend to actively track feature requests and bug reports using our Github repository.

8.5 Baseline Models on VANiLLa

This section presents an overview of our baseline models used for evaluating and assessing the quality of the data. We decided to use some conventional sequence-to-sequence models following the underlying Encoder-Decoder pipeline [41, 42], as shown in Figure 8.4. The question Q_{NL} and the retrieved answer A_T are separated with the separation "<*sep*>" tag to form the input to the encoder and the decoder produces the verbalized answer in natural language A_V as the output. The inputs to both the encoder and the decoder end with the end-of-sequence "<*eos*>" tag and the output from the decoder is appended with the start-of-sequence "<*sos*>" tag before the start of prediction. The four baseline models are as follows:

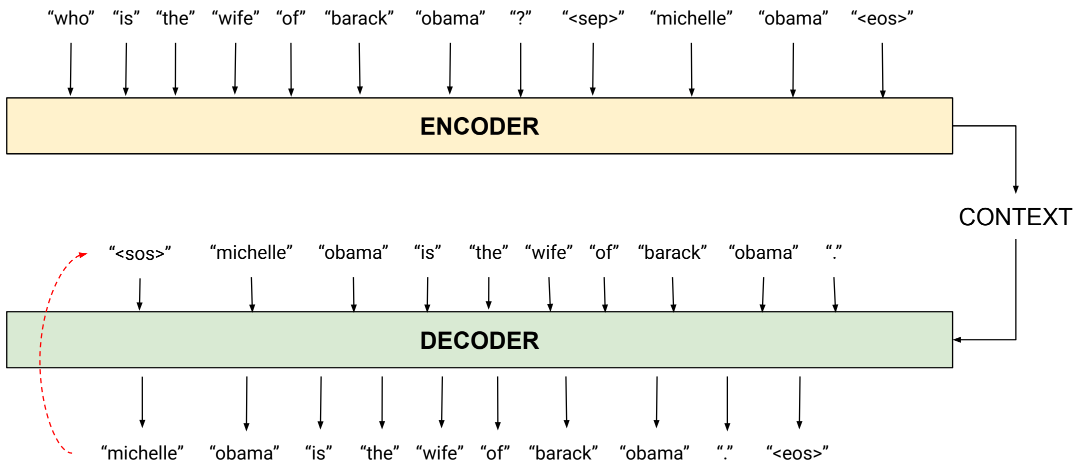

"who" "is" "the" "wife" "of" "barack" "obama" "?" "<sep>" "michelle" "obama" "<eos>"
ENCODER
CONTEXT
"<sos>" "michelle" "obama" "is" "the" "wife" "of" "barack" "obama" "."
DECODER
"michelle" "obama" "is" "the" "wife" "of" "barack" "obama" "." "<eos>"

Figure 8.4: Basic architecture of the baseline models

BL1: Sequence-to-Sequence model with attention mechanism

Seq2Seq model with attention mechanism [43] proposed by Bahdanau simultaneously learns to align and translate by employing a bi-directional RNN as the encoder and using a weighted sum of the vector representation generated at the encoder level as the context in the decoder. These sequence-to-sequence

[3]https://figshare.com/articles/Vanilla_dataset/12360743
[4]https://creativecommons.org/licenses/by/4.0/
[5]https://github.com/AskNowQA/VANiLLa
[6]https://www.gnu.org/licenses/gpl-3.0.html

models have been widely used in the NLP community for tasks such as Machine Translation (MT) and Question Answering (QA).

BL2: Convolution based Encoder-Decoder model

Facebook AI introduced the use of CNN in sequence learning [117]. The concept is to implement a fully convolutional architecture, by incorporating CNN at both encoder and decoder level. These models implement a position embedding, which provides an understanding about the portion of input or output sequence currently in consideration, along with a multi-step attention.

BL3: Transformer

Transformers are novel architectures that solves sequence-to-sequence tasks quite nicely because of its ability to handle long-term dependencies. Recently, Transformers achieved state-of-the art in wide range of NLP tasks such as Natural Language Generation (NLG), Machine Translation (MT). The architecture of transformer was first proposed by Vaswani et al. [118]. Followed by that many pre-trained transformer models are proposed such as BERT [165], GPT [166], RoBERTa [167], which are trained on a large number of data.

For simplicity, we use a simple transformer model for the evaluation of our dataset. Multi-headed attention and positional-encoding are used in this transformer model. Multi-headed attention helps to capture global dependencies between input and output and positional-encoding encodes information about the relative or absolute position of each token in the sequence.

8.6 Evaluation

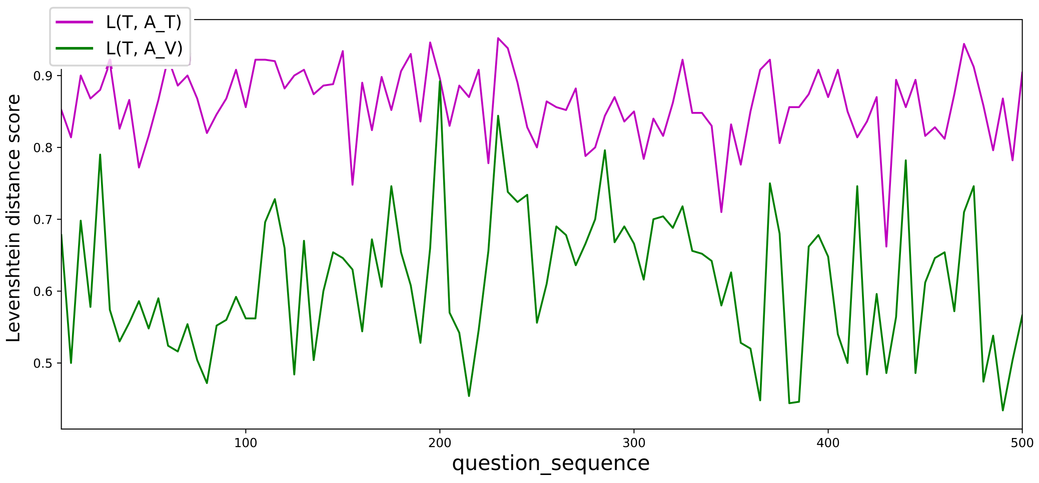
In this section, we report the different experiments we have performed for the assessment of the quality of our dataset and for comparing the performance of our baseline models.

8.6.1 Experiment 1: Comparing Similarity Measures

Objective: A more human-like response to a user question Q_{NL} would mean that the generated response in natural language provides some context of the question. From this point of view, we perform some experiments to show that the verbalized answers A_V in our dataset preserves the semantic and syntactic meaning of the question more compared to the relevant RDF triples verbalization A_T. Additionally, we also indicate that a response generated by verbalizing the RDF triple A_T would be semantically and syntactically more close to the T in contrast to the that of A_V and T. Concisely, we evaluate hypothesis **H1** from section 8.1.

Experimental Settings: We randomly sample 500 questions from our dataset to perform this experiment. We generated the verbalization of the RDF triples associated with these questions with the help of a simple AMT experiment where the turkers were asked to rewrite the triples in the form of natural language sentences without any information about the questions.

To analyze the quality of our dataset, we calculate the similarity measure Sim: (i) as the Levenshtein distance using Fuzzywuzzy, denoting the "syntactic similarity" and (ii) as the cosine vector similarity using BERT embeddings, denoting the "semantic similarity", of T and Q_{NL} with both A_V and A_T. We record the mean μ and standard deviation σ for all the comparisons over the randomly sampled data.



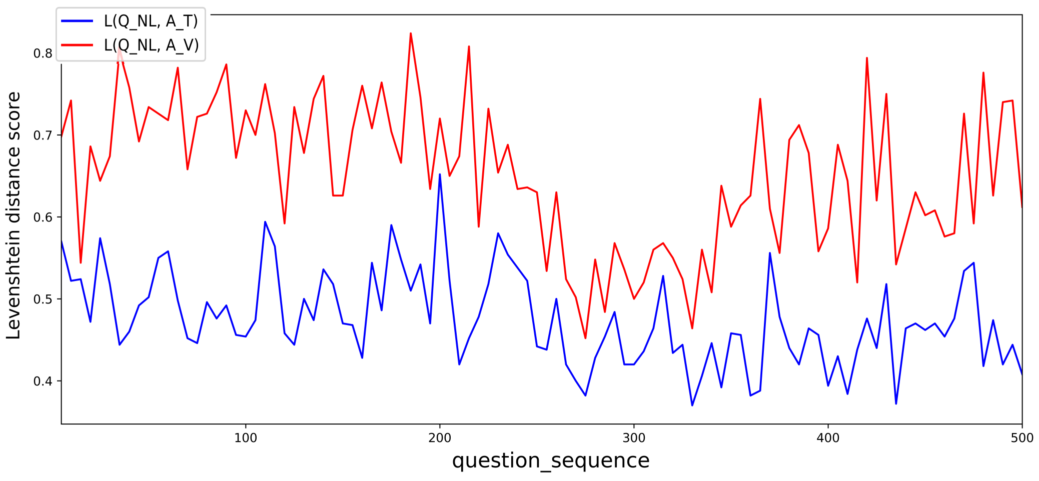


Figure 8.5: Syntactic and semantic comparison of T and Q_{NL} with both A_V and A_T

Experimental Results: The results of our experiment are depicted in Table: 8.2. On the basis of Levenshtein distance, we observe that the syntactic similarity between the verbalized RDF triple A_T and the RDF triple T is more than the similarity between T and the verbalized answer A_V, i.e., $Sim(A_T, T) > Sim(A_V, T)$. In contrary, the syntactic similarity between A_V and the user question Q_{NL} is more than the similarity Q_{NL} and the verbalized RDF triple A_T, i.e., $Sim(A_V, Q_{NL}) > Sim(A_T, Q_{NL})$. We also observe the same in the case of the cosine similarity measure. Thus, empirically proving hypothesis **H1**.

Additionally, Figure: 8.5 illustrates The results of the experiment calculating the Levenshtein distances. In the first plot, we compare the Levenshtein distance between (T, A_T) and (T, A_V) over the 500 randomly sampled questions. The plot clearly depicts that the purple line indicating the distance between (T, A_T) is higher than the green line indicating the distance between (T, A_V) at most data point, thus demonstrating that the pair (T, A_V) is syntactically less similar than the (T, A_T). Furthermore in the second plot, we compare the Levenshtein distance between (Q_NL, A_T) and (Q_NL, A_V), which clearly depicts the red line indicating the distance between (Q_{NL}, A_V) is higher than the blue line indicating the distance between (Q_{NL}, A_T). Thus, verifying that A_V is syntactically more similar to Q_{NL} than A_T. However, in both case

		Levenshtein Ratio		**Cosine-Similarity**	
		$\mathbf{A_T}$	$\mathbf{A_V}$	$\mathbf{A_T}$	$\mathbf{A_V}$
T	μ	0.8602	0.6139	0.8441	0.7965
	σ	0.1066	0.1779	0.0793	0.0992
$\mathbf{Q_{NL}}$	μ	0.4750	0.6519	0.3528	0.3574
	σ	0.1018	0.1504	0.1064	0.1056

Table 8.2: Comparing A_V, A_T, Q_{NL} and T based on semantic cosine vector and syntactic Levenshtein distance similarity

there are a few points of exception where the similarity measures are the same. This is due to questions in the dataset which contain same surface labels as in the RDF triples.

8.6.2 Experiment 2: Evaluating Baseline Models on VANiLLa Dataset

Objective: The objective of this experiment is to record the performance of our baseline models on VANiLLa dataset based on certain evaluation criteria.

Dataset: We use our dataset VANiLLa with a 80-20 split to generate the training and test sets. The training set consists of 85732 examples whereas the test set consists of 21434 examples. We also implement a k-cross validation with $k = 5$ on training set for generating the validation set in both cases.

Experimental Settings: We use standard dropout value of 0.5 for the recurrent layers, 0.25 for convolution layers and 0.1 for multi-head attention mechanism. The hidden layers dimensions for transformers is set to 200 with embedding dimension of 200, while for the other baselines are set the hidden dimension to 512 and embedding dimension to 300. In the case of BL2, we utilize 10 layers of convolutions with kernel size 3. For the multi-headed attention mechanism in BL3, we set 8 as the number of head with 4 layers. For the sake of consistency, we train all the models for 7 epochs on the our dataset, with Adam optimizer and cross entropy loss function.

Evaluation Criteria: We implement the following criteria for the evaluation and comparison of our baseline models:

Criteria#1 **Perplexity**: Perplexity defines the quality of predicting real sentences for a probability model. A good model will have a lower perplexity indicating a high probability of prediction. It is the average branching factor for predicting the next word. It is calculated with the help of cross-entropy as follows:

$$PPL = 2^{H(p,q)} \tag{8.3}$$

where, $H(p, q)$ is the cross-entropy of the model which tries to learn the probability distribution q close to the unknown probability distribution of the data p.

Criteria#2 **Modified Precision**: Modified Precision is the sum of the clipped n-gram counts of the generated sequence or hypothesis divided by the total number of n-grams in the hypothesis. The range of the modified precision is between 100 and 0 with 100 being the best score. This scores depicts two aspects: (i) adequacy (generated hypothesis sentence contains similar words as that in the reference sentence), and (ii) fluency (the n-gram matches between the hypothesis and reference).

Criteria#3 **BLEU**: The BLEU score, Bilingual Evaluation Understudy, is a modified precision score [168] combined with a brevity penalty for penalizing shorter generation. Hence, the BLEU score checks for matches in length, order and word choice between the hypothesis and the references. The range of BLEU score is between 100 and 0 with 100 being the best score.

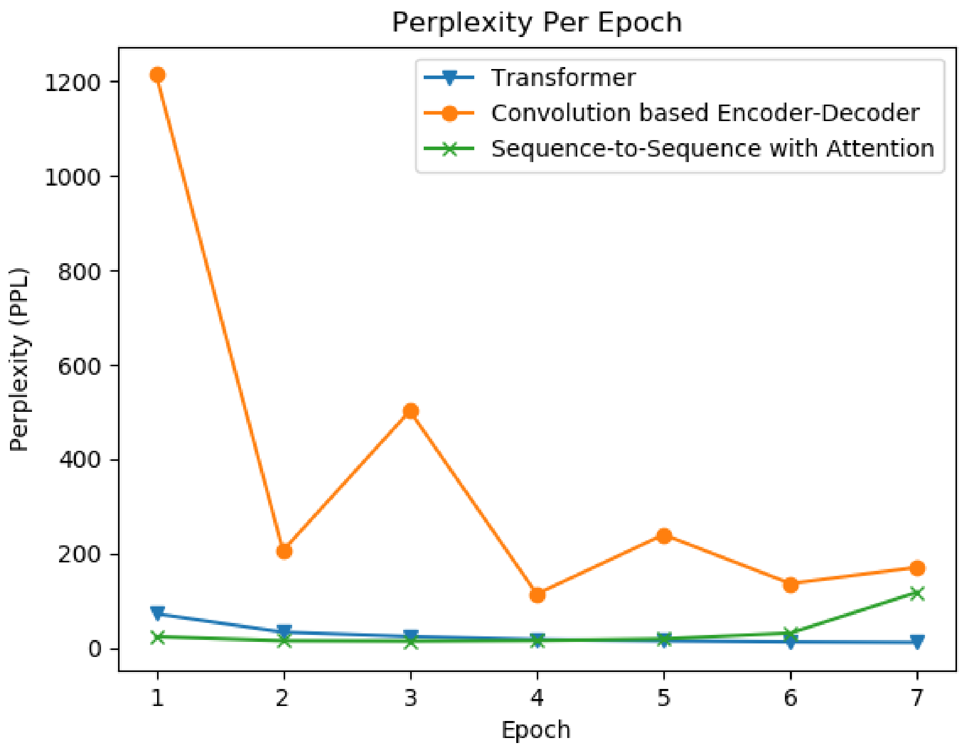


Figure 8.6: Perplexity(PPL) per Epoch during training

Experimental Results: The perplexity value on the validation set after every epoch for all our baselines are plotted in Figure: 8.6 and the test results of the three baseline models trained on the two sizes of our dataset, VANiLLa has been reported in Table 8.3. The results clearly indicate that the transformer model performs the best among the three baselines with perplexity value of 12.10 and BLEU score of 30.80.

Baseline Model	PPL	Precision	BLEU
BL1	27.91	19.84	16.66
BL2	87.67	70.50	15.42
BL3	**12.10**	**76.00**	**30.80**

Table 8.3: Evaluating Baseline Models on VANiLLa Dataset (No. of Epochs: 7)

The transformer model puts attention on the whole sentence instead of attention on word by word, hence capturing the context better. Additionally, transformer handles the relationship between words in a sentence by positional encoding and multi-headed attention mechanism [118]. These characteristics helps transformer model achieve better result than other baseline models.

8.6.3 Experiment 3: Combining VQuAnDa and VANiLLa Datasets

Objective: Empirically it is evident that the transformer model is the best amongst the baseline models. Hence, in this experiment we examine the improvement of the evaluation metrics for the transformer model trained on the combination of VANiLLa and VQuAnDa [115] datasets (5k question-answer pairs).

Training Set	Test Set	PPL	Precision	BLEU
VANiLLa + VQuAnDa	VQuAnDa	132.26	**80.45**	21.61
	VANiLLa	9.05	73.63	**35.70**
VQuAnDa	VQuAnDa	84.88	75.05	26.25

Table 8.4: Evaluating Baseline Model BL3 on VANiLLa and VQuAnDa Dataset (No. of Epochs 50)

Experimental Settings: In this experiment we combine the training data of VANiLLa with the training data of VQuAnDa dataset and feed it into the transformer model. Although, we combine the training data for the training phase but we test the model on both the VANiLLa and VQuAnDa separately. Note that the majority of the combined data is from VANiLLa as the number of data in VANiLLa is 20 times more than VQuAnDa.

Experimental Results: The combined data used for training improves the precision score (80.25) of VQuAnDa test data by a fair margin. Additionally, transformer model achieves best BLEU score on VANiLLa test set. So, it is evident that this combined data training approach helps in creating a strong baseline for future research. Furthermore, our baseline transformer model with the suggested parameters (described in the experimental settings) achieve the state-of-the-art in BLEU score on VQuAnDa data which is 26.25 where as VQuAnDa reports BLEU is reported 18.18.

8.7 Conclusion and Future Work

In this chapter, we work towards improving the answer representation of the KGQA system by curating an answer verbalization dataset. We introduce VANiLLa dataset with the intent to support the verbalization of answers in a natural language task. Here the verbalized answer imbibes the context from the question by using parts of the input user questions. The dataset consisting of over 100k examples was generated semi-automatically by adapting questions from other datasets such as CSQA and SimpleQuestionsWikidata and the verbalizing the answers by crowd-sourcing. We also provide baseline models based on the generic sequence to sequence models already in use in the NLP community. These baseline models present a reference point for future research. The experimental results of these baselines depict that there still exists a lot of room for enhancement until we solve the task of verbalizing answers. In future, we hope to continuously aid researchers with more variations and expansions of our datasets and also to built better methodologies to tackle the problem.

CHAPTER 9

Conclusion and Future Directions

In this thesis, we follow the research objective of improving state of the art of the KGQA for Complex questions. We defined the research problem in Chapter 1 and discussed the significant challenges to overcome in order to achieve the research objective. In Chapter 2, we discussed all the fundamentals and necessary background concepts required for this thesis. Chapter 3 covers all the related work according to the research objective and research question. To achieve our research objective and overcome the research challenges, we have broken down the research problem to five research questions. We tackled these research questions in the subsequent five chapters of the thesis.

In the following sections, we provide a summary of our research questions and the contributions towards them, elaborating main findings that validate our research questions.

9.1 Review of the Contributions

In this section, we summarize the contribution provided in the thesis from Chapter 4, 5, 6, 7, 8. Individually we first state the research question and then the contributions towards them.

Research Question 1 (RQ1)

How does the semantic understanding of the user question affect the performance of KGQA?

Understanding the semantics of a natural language question plays a critical role in question answering. Semantics have an even more important role in complex questions as here the structural and syntactical variation is substantial both at natural language and formal expression. We address the research question RQ1 by developing Normalized Query Structure (NQS) that captures the semantics of the natural language and eases the process of formal query(SPARQL in this case) generation. NQS acts like an intermediate state between natural language question and formal language query. We developed a slot filling algorithm NL-to-NQS, based on POS tag and NER to annotate user question with user desire(intent), user input and the relation between them. As natural language could be paraphrased and yet hold the same semantics, so we developed NQS in such a way that it could capture the same semantics from the set of paraphrased questions. Further, we developed NQS-to-SPARQL module such that we could generate a SPARQL query based on NQS instance. We achieved near state of the art score on the QALD-5 dataset.

Contributions to Research Question RQ1 are summarized as follows:

- A novel paraphrase resilient query characterization structure (and algorithm), called *NQS*(Normalized Query Structure), is proposed. NQS is less sensitive to structural variation. It supports complex queries, hence, serves as a robust intermediary formal query representation.

- An NQS to SPARQL translation algorithm (and tool) is proposed, that supports user queries to be agnostic of the target RDF store structure and vocabulary.

- An evaluation of AskNowNQS in terms of: (i) assessing robustness using the Microsoft Encarta data set, and (ii) evaluating accuracy using a community query dataset built on top of OWL-S TC v.4.0 and QALD-4/5 datasets.

Research Question 2 (RQ2)

How to generate a large size Complex question dataset for KGQA, with a wide variety of questions, so the Machine Learning could leverage over the dataset?

To achieve our research objective, we find a potential challenge of not having an extensive size data set with the complex question for KGQA. All the previous published datasets were either small in size or were towards simple questions. To apply machine learning over complex questions, we need a large scale dataset with a wide variety of questions and formal query patterns. To overcome this challenge, we establish LC-QuAD framework which allowed us to curate large size KGQA dataset, with reduced efforts from the domain experts. The LC-QuAD framework reverts the process of question answering and generates the natural language question from the automatically generated SPARQL. We handpick the entity from DBpedia and extract 2-hop subgraph. We prune the subgraph based on the white-list of predicates created by us. Finally, we had several different SPARQL templates, and SPARQL queries were created based on the pruned sub-graph. We translate these SPARQL queries by using a template-based translation, which generated natural questions which were clear to humans. Then we manually correct the grammar of these question and add more variety to the template-based translation output by paraphrasing the question. At the end, during review process, we check the final quality of dataset generated. After publishing the LC-QuAD 1.0 dataset, we have seen the impact in the research community as there are several KGQA baseline and modules being published using this complex question dataset.

Contributions to Research Question RQ2 are summarized as follows:

- A framework named LC-QuAD (Large-Scale Complex Question Answering Dataset), for generating NLQs and their SPARQL queries which reduces the need for manual intervention.

- A dataset LC-QuAD 1.0, of 5000 questions with their intended SPARQL queries for DBpedia 2016-04. The questions exhibit large syntactic and structural variations.

- We created other variations of the LC-QuAD 1.0 dataset by making the dataset available with full annotation and also releasing German version the dataset.

Research Question 3 (RQ3)

How to incorporate Reification of Knowledge Graph in KGQA dataset so that the QA system utilize it?

Reified Knowledge Graphs are the KGs which contextualize the metadata related to a triple by adding a property-value pair. Such KGs can procure even more complex facts and potentially have more detailed information. For instance, a fact in Reified KG could have details of temporal reference points in the form of metadata. Acquiring this qualifier based information in natural language question would make the questions more informative, and the user could ask more precious information. To the best of our knowledge, KGQA has never explored the qualifier based KGs. Thus we aim to introduce a large scale dataset which uses reified KGs, and the SPARQL uses the qualifier based metadata, thus curating more complex and informative question. For this task, we extend the LC-QuAD framework in areas, first by adopting the SPARQL generation to qualifier based KGs (Wikidata and DBpedia 2018), second by up-scaling the question verbalization process by using crowd-sourcing.

Contributions to Research Question RQ3 are summarized as follows:

- Provision of the largest dataset of 30,000 complex questions with corresponding SPARQL queries for Wikidata and DBpedia 2018.
- All questions in LCQuAD 2.0 also consist of paraphrased versions via crowd-sourcing tasks. This provide more natural language variations for the question answering system to learn from and avoids over-fitting on a small set of syntactic variations.
- Questions in this dataset have a good variety and complexity levels such as multi-fact questions, temporal questions and questions that utilize qualifier information.
- This is the first KGQA dataset which contains questions with dual user intents and questions that require SPARQL string operations .

Research Question 4 (RQ4)

How effective is the joint task of Entity Linking task and Relation Linking for KGQA?

Entity Linking and Relation Linking are the critical task for the KGQA. Low accuracy in these components hampers the performance of the KGQA. The Entity Linking systems commonly used in KGQA are not explicitly designed for questions, instead, most Entity Linking are trained for large texts which consist of multiple entities. Complex questions mostly have one or two entities, and this is not an optimal setting for traditional entity linkers. On the other side, KGQA questions have relations in the close vicinity of the entities of the question. Thus, we propose to perform the two tasks of Entity linking and Relation linking as a joint task. We developed an Entity and Relation Linking module for KGQA based on the LC-QuAD 1.0 dataset. We explored two different methodologies to perform this joint task, first based on the Generalized Traveling Salesman Problem and second based on Connection Density between the nodes in the KG. We achieved the state of the art results for Entity Linking and

Relation Linking over LC-QuAD 1.0 and QALD 9 dataset, clearly showcasing the advantage of joint task approach.

Contributions to Research Question RQ4 are summarized as follows:

- The framework EARL, where GTSP solver or Connection Density can be used for joint linking of entities and relations.
- A formalization of the joint entity and relation linking problem as an instance of the Generalized Traveling Salesman (GTSP) problem .
- An implementation of the GTSP strategy using approximate GTSP solvers.
- A "Connection Density" formalization and implementation of the joint entity and relation linking problem as a machine learning task.
- An adaptive E/R learning module, which can correct errors occurring across different modules.
- A large set of labels for DBpedia predicates and entities covering the syntactic and semantic variations.

Research Question 5 (RQ5)

How to improve the answer representation by using the context and textual description from the user question?

The last challenge we address in this thesis to achieve our research objective is to improve the answer representation for enhanced communication between the user and the KGQA system. While showcasing the answer to the user, most systems either directly show the labels of the retrieved answer, some systems use a template-based technique on triple verbalization. We improve the answer representation by verbalizing the answer based on the textual information and context from the user question. Our goal is to provide response-answers which mimic more human-like response to the user question. To achieve this, we first create a large-scale dataset question with the verbalized answer by extracting question from two public datasets. We then applied several deep learning models to develop baselines for our datasets.

Contributions to Research Question RQ5 are summarized as follows:

- Provision of a large-scale dataset of over 100k questions along with their corresponding verbalized answers.
- We present a cost-efficient semi-automatic framework for generating large-scale answer verbalization datasets, which was also used to generate VANiLLa.
- We provide multiple baseline models based on the conventional sequence-to-sequence framework as reference for future research.
- We also provide empirical proof that answer verbalizations are better suited response and provide more context compared to RDF triple verbalizations.

9.2 Limitations and Future Directions

Despite the overall achieved research objective, there are few limitations of this research which have not been covered in the scope of the thesis. We list the following limitations:

- The NQS presented in chapter 4 is designed for KGQA questions and remains untested on general reasoning or procedural questions. Moreover, its rule-based architecture and the coverage of the NQS is completely limited to the rules added based on the linguistic heuristics and observing patterns in the data.
- The current implementation of LC-QuAD framework is KG schema oriented, though it is reasonably extensible to other KGs too, as we did showcase in chapter 6. The final Natural Language Questions(NLQ) in these datasets are biased towards the underline template, as the generation of these Questions is based on NNQT templates. In LC-QuAD 2.0 (Chapter 6), we overcome this issue by using the paraphrasing of every NLQ independent of NNQT, to get more variation in the Natural Language Question.
- The EARL framework relies on span detection for entity and relation linking; thus, EARL can only identify explicit relations and will miss-out any hidden relations in the question. Further errors in span detection are cascaded further in the pipeline and results in drop of accuracy.
- The answer verbalization dataset from chapter 8 entirely relies on interpretation of the question and answer formulation of the crowd-source workers; thus, the vocabulary will be limited.

Based on our findings, and the contributions made in this thesis, we now present some of the future directions for the research community:

- The current implementation of NQS is developed on rules based on POS (parts of speech) tags and NER (named entity recognition) tags. In future, one could deploy machine learning, particularly models like Seq2Seq, to learn the patterns from the data instead then manual-rules. This would improve the accuracy and coverage of NQS and make NQS more generalized.
- The question generation process developed in this thesis has drastically reduced manual efforts of domain experts to curate a large dataset. A future direction would be to have fully Automatic Question Generation for Complex Question and automatic paraphrasing of question using deep learning techniques.
- The proposed Entity and Relation linking system EARL uses graph properties and classical graph algorithm to perform the linking task. Perhaps one could explore further in this direction and extend the approach and try to retrieve answers of the questions without formal query construction and just based on graph properties and classical algorithm. This approach could potentially be beneficial for cases where not much training data is available. Another area of improvement could also be to consider the whole question while re-ranking in the final stage of EARL.
- The answer verbalization process discussed in this thesis (Chapter 8) is based on the information from the question and KG triples. Moving towards Reified KGs, we could have more informative answer verbalization by using the qualifier information.

9.3 Closing Remarks

Knowledge Graph research is continuously evolving, and they are becoming more capable of storing accurate and sophisticated information in an interconnected fashion. Question Answering over Knowledge Graphs with the complex question is getting more attention and will keep getting evolved in future.

During this thesis, we advanced the state of the art in complex question answering on several fronts by setting up benchmarks and developing modules dedicated to KGQA. More specifically, we proposed:

- efficient and scalable approach to generate KGQA dataset,
- an intermediate state to ease the process of query formalization,
- Another KGQA module to perform entity and relation linking task with state of the art results and
- We explored the field of answer verbalization for improving the system answer quality.

Future research work can build upon the datasets and modules developed as contributions presented during this thesis. These contributions could provide a foundation for complex question answering system with high accuracy. Furthermore, contributions of this thesis are already making an impact on the question of answering community, as several other projects are working on the KGQA datasets and modules published within the thesis duration. We integrated all contribution of the thesis within the umbrella Project AskNow.

Bibliography

[1] B. F. Green, 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,
IRE-AIEE-ACM '61 (Western),
Los Angeles, California: Association for Computing Machinery, 1961 219,
ISBN: 9781450378727, URL: `https://doi.org/10.1145/1460690.1460714`
(cit. on p. 1).

[2] W. A. Woods,
"Lunar Rocks in Natural English: Explorations in Natural Language Question Answering",
Linguistic Structures Processing, Amsterdam: North Holland, 1977 521 (cit. on p. 1).

[3] C. C. T. Kwok, O. Etzioni and D. S. Weld, *Scaling question answering to the web*,
ACM Trans. Inf. Syst. **19**.3 (2001) 242,
URL: `https://doi.org/10.1145/502115.502117` (cit. on p. 1).

[4] 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 World Wide Web conference, 2012 639 (cit. on pp. 1, 5).

[5] 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",
The Semantic Web - ISWC 2019 - 18th International Semantic Web Conference, Auckland, New Zealand, October 26-30, 2019, Proceedings, Part I, vol. 11778,
Lecture Notes in Computer Science, Springer, 2019 487,
URL: `https://doi.org/10.1007/978-3-030-30793-6%5C_28`
(cit. on pp. 1, 40, 79).

[6] K. Singh, A. S. Radhakrishna, A. Both, S. Shekarpour, I. Lytra, R. Usbeck, A. Vyas,
A. Khikmatullaev, D. Punjani, C. Lange, M. Vidal, J. Lehmann and S. Auer,
"Why Reinvent the Wheel: Let's Build Question Answering Systems Together",
Proceedings of the 2018 World Wide Web Conference on World Wide Web, WWW 2018, Lyon, France, April 23-27, 2018, ACM, 2018 1247,
URL: `https://doi.org/10.1145/3178876.3186023` (cit. on pp. 2, 24, 46, 80, 96).

[7] M. Petrochuk and L. Zettlemoyer,
"SimpleQuestions Nearly Solved: A New Upperbound and Baseline Approach",
Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing,
Brussels, Belgium: Association for Computational Linguistics, 2018 554,
URL: `https://www.aclweb.org/anthology/D18-1051` (cit. on p. 2).

[8] M. Färber, F. Bartscherer, C. Menne and A. Rettinger, *Linked data quality of DBpedia, Freebase, OpenCyc, Wikidata, and YAGO*, Semantic Web **9**.1 (2018) 77, URL: `https://doi.org/10.3233/SW-170275` (cit. on pp. 2, 36).

[9] H. Thakkar, K. M. Endris, J. M. Giménez-Garcıa, J. Debattista, C. Lange and S. Auer, "Are Linked Datasets fit for Open-domain Question Answering? A Quality Assessment", *Proceedings of the 6th International Conference on Web Intelligence, Mining and Semantics, WIMS 2016, Nımes, France, June 13-15, 2016*, ACM, 2016 19:1, URL: `https://doi.org/10.1145/2912845.2912857` (cit. on p. 2).

[10] A. Bordes, N. Usunier, S. Chopra and J. Weston, *Large-scale Simple Question Answering with Memory Networks*, CoRR **abs/1506.02075** (2015) (cit. on pp. 4, 5, 43, 70, 71, 88).

[11] I. V. Serban, A. García-Durán, Ç. Gülçehre, S. Ahn, S. Chandar, A. Courville and Y. Bengio, "Generating Factoid Questions With Recurrent Neural Networks: The 30M Factoid Question-Answer Corpus", *54th Annual Meeting of the Association for Computational Linguistics*, 2016 (cit. on pp. 4, 41, 43, 47, 70, 88).

[12] W.-t. Yih, M.-W. Chang, X. He and J. Gao, "Semantic Parsing via Staged Query Graph Generation: Question Answering with Knowledge Base", *Proceedings of the 53rd Annual Meeting of the ACL and the 7th International Joint Conference on NLP*, 2015 (cit. on pp. 4, 43, 70, 71, 88).

[13] R. Usbeck, R. H. Gusmita, A. N. Ngomo and M. Saleem, "9th Challenge on Question Answering over Linked Data (QALD-9) (invited paper)", *Joint proceedings of the 4th Workshop on Semantic Deep Learning (SemDeep-4) and NLIWoD4: Natural Language Interfaces for the Web of Data (NLIWOD-4) and 9th Question Answering over Linked Data challenge (QALD-9) co-located with 17th International Semantic Web Conference (ISWC 2018), Monterey, California, United States of America, October 8th - 9th, 2018*, vol. 2241, CEUR Workshop Proceedings, CEUR-WS.org, 2018 58, URL: `http://ceur-ws.org/Vol-2241/paper-06.pdf` (cit. on pp. 4, 43, 88, 115).

[14] J. Berant, A. Chou, R. Frostig and P. Liang, "Semantic Parsing on Freebase from Question-Answer Pairs.", *EMNLP*, vol. 2, 2013 6 (cit. on pp. 5, 42, 43, 70, 96).

[15] 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. 10, 52, 71, 83, 96).

[16] 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. 11, 70, 88, 103).

[17] M. Dubey, D. Banerjee, A. Abdelkawi and J. Lehmann, "LC-QuAD 2.0: A Large Dataset for Complex Question Answering over Wikidata and DBpedia", *The Semantic Web - ISWC 2019 - 18th International Semantic Web Conference, Auckland, New Zealand, October 26-30, 2019, Proceedings, Part II*, vol. 11779, Lecture Notes in Computer Science, Springer, 2019 69,

URL: https://doi.org/10.1007/978-3-030-30796-7%5C_5
(cit. on pp. 12, 84, 115).

[18] 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, ACM, 2011 1
(cit. on pp. 12, 44, 94, 106).

[19] R. Usbeck, A.-C. N. Ngomo, M. Röder, D. Gerber, S. A. Coelho, S. Auer and A. Both,
"AGDISTIS-graph-based disambiguation of named entities using linked data",
International Semantic Web Conference, Springer, 2014 457 (cit. on pp. 12, 44, 94, 106).

[20] M. Dubey, D. Banerjee, D. Chaudhuri and J. Lehmann,
"Earl: Joint entity and relation linking for question answering over knowledge graphs",
International Semantic Web Conference, Springer, 2018 108 (cit. on pp. 12, 80, 85, 94).

[21] T. Berners-Lee and M. Fischetti,
Weaving the web - the original design and ultimate destiny of the World Wide Web by its inventor,
HarperBusiness, 2000, ISBN: 978-0-06-251587-2 (cit. on p. 15).

[22] L. Ehrlinger and W. Wöß, "Towards a Definition of Knowledge Graphs",
Joint Proceedings of the Posters and Demos Track of the 12th International Conference on Semantic Systems - SEMANTiCS2016 and the 1st International Workshop on Semantic Change & Evolving Semantics (SuCCESS'16) co-located with the 12th International Conference on Semantic Systems (SEMANTiCS 2016), Leipzig, Germany, September 12-15, 2016, vol. 1695,
CEUR Workshop Proceedings, CEUR-WS.org, 2016,
URL: http://ceur-ws.org/Vol-1695/paper4.pdf (cit. on p. 17).

[23] Q. Wang, Z. Mao, B. Wang and L. Guo,
Knowledge Graph Embedding: A Survey of Approaches and Applications,
IEEE Trans. Knowl. Data Eng. **29**.12 (2017) 2724,
URL: https://doi.org/10.1109/TKDE.2017.2754499 (cit. on p. 17).

[24] J. Pérez, M. Arenas and C. Gutiérrez, *Semantics and complexity of SPARQL*,
ACM Trans. Database Syst. **34**.3 (2009) 16:1,
URL: https://doi.org/10.1145/1567274.1567278 (cit. on pp. 18, 25, 28).

[25] A. Gubichev and M. Then, "Graph Pattern Matching: Do We Have to Reinvent the Wheel?",
Proceedings of Workshop on GRAph Data, ACM, 2014 (cit. on p. 18).

[26] S. Ji, S. Pan, E. Cambria, P. Marttinen and P. S. Yu,
A Survey on Knowledge Graphs: Representation, Acquisition and Applications,
CoRR **abs/2002.00388** (2020), arXiv: 2002.00388,
URL: https://arxiv.org/abs/2002.00388 (cit. on pp. 18, 19).

[27] K. Purcell, L. Rainie and J. Brenner, *Search engine use 2012*, (2012) (cit. on p. 19).

[28] A. Rajaraman and J. D. Ullman, "Data Mining", *Mining of Massive Datasets*,
Cambridge University Press, 2011 1 (cit. on p. 19).

[29] X. Ling, S. Singh and D. S. Weld, *Design Challenges for Entity Linking*,
Trans. Assoc. Comput. Linguistics **3** (2015) 315, URL: https://tacl2013.cs.columbia.edu/ojs/index.php/tacl/article/view/528
(cit. on p. 22).

[30] D. Rao, P. McNamee and M. Dredze,
"Entity Linking: Finding Extracted Entities in a Knowledge Base",
Multi-source, Multilingual Information Extraction and Summarization,
Theory and Applications of Natural Language Processing, Springer, 2013 93,
URL: `https://doi.org/10.1007/978-3-642-28569-1%5C_5` (cit. on p. 22).

[31] E. Marx, T. Soru, D. Esteves, A. N. Ngomo and J. Lehmann,
"An Open Question Answering Framework",
Proceedings of the ISWC 2015 Posters & Demonstrations Track co-located with the 14th International Semantic Web Conference (ISWC-2015), Bethlehem, PA, USA, October 11, 2015,
vol. 1486, CEUR Workshop Proceedings, CEUR-WS.org, 2015,
URL: `http://ceur-ws.org/Vol-1486/paper%5C_1.pdf` (cit. on p. 24).

[32] J. Kim, C. Unger, A. N. Ngomo, A. Freitas, Y. Hahm, J. Kim, G. Choi, J. Kim, R. Usbeck, M. Kang and K. Choi, "OKBQA: an Open Collaboration Framework for Development of Natural Language Question-Answering over Knowledge Bases", *Proceedings of the ISWC 2017 Posters & Demonstrations and Industry Tracks co-located with 16th International Semantic Web Conference (ISWC 2017), Vienna, Austria, October 23rd - to - 25th, 2017*, vol. 1963,
CEUR Workshop Proceedings, CEUR-WS.org, 2017,
URL: `http://ceur-ws.org/Vol-1963/paper616.pdf` (cit. on p. 24).

[33] D. Hernández, A. Hogan and M. Krötzsch, "Reifying RDF: What Works Well With Wikidata?", *Proceedings of the 11th International Workshop on Scalable Semantic Web Knowledge Base Systems co-located with 14th International Semantic Web Conference (ISWC 2015), Bethlehem, PA, USA, October 11, 2015*, vol. 1457, CEUR Workshop Proceedings, CEUR-WS.org, 2015 32,
URL: `http://ceur-ws.org/Vol-1457/SSWS2015%5C_paper3.pdf`
(cit. on p. 27).

[34] J. Frey, K. Müller, S. Hellmann, E. Rahm and M. Vidal,
Evaluation of metadata representations in RDF stores, Semantic Web **10**.2 (2019) 205,
URL: `https://doi.org/10.3233/SW-180307` (cit. on p. 27).

[35] O. Hartig, "Foundations of RDF⋆ and SPARQL⋆ (An Alternative Approach to Statement-Level Metadata in RDF)", *Proceedings of the 11th Alberto Mendelzon International Workshop on Foundations of Data Management and the Web, Montevideo, Uruguay, June 7-9, 2017*, vol. 1912, CEUR Workshop Proceedings, CEUR-WS.org, 2017,
URL: `http://ceur-ws.org/Vol-1912/paper12.pdf` (cit. on pp. 27, 28).

[36] O. Hartig, "RDF* and SPARQL*: An Alternative Approach to Annotate Statements in RDF", *Proceedings of the ISWC 2017 Posters & Demonstrations and Industry Tracks co-located with 16th International Semantic Web Conference (ISWC 2017), Vienna, Austria, October 23rd - to - 25th, 2017*, vol. 1963, CEUR Workshop Proceedings, CEUR-WS.org, 2017,
URL: `http://ceur-ws.org/Vol-1963/paper593.pdf` (cit. on p. 27).

[37] O. Hartig and B. Thompson, *Foundations of an Alternative Approach to Reification in RDF*, CoRR **abs/1406.3399** (2014), arXiv: `1406.3399`,
URL: `http://arxiv.org/abs/1406.3399` (cit. on p. 28).

[38] S. Santhanam and S. Shaikh, *A Survey of Natural Language Generation Techniques with a Focus on Dialogue Systems - Past, Present and Future Directions*, CoRR **abs/1906.00500** (2019), arXiv: `1906.00500`, URL: `http://arxiv.org/abs/1906.00500` (cit. on p. 30).

[39] J. Gao, M. Galley and L. Li, *Neural Approaches to Conversational AI*, CoRR **abs/1809.08267** (2018), arXiv: `1809.08267`, URL: `http://arxiv.org/abs/1809.08267` (cit. on p. 30).

[40] E. Reiter and R. Dale, *Building Natural Language Generation Systems*, 2000 (cit. on p. 30).

[41] I. Sutskever, O. Vinyals and Q. V. Le, "Sequence to sequence learning with neural networks", *Advances in neural information processing systems*, 2014 3104 (cit. on pp. 32, 119).

[42] 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*, 2014, arXiv: `1406.1078 [cs.CL]` (cit. on pp. 32, 119).

[43] D. Bahdanau, K. Cho and Y. Bengio, *Neural Machine Translation by Jointly Learning to Align and Translate*, 2014, arXiv: `1409.0473 [cs.CL]` (cit. on pp. 33, 49, 50, 119).

[44] T. Mikolov, K. Chen, G. Corrado and J. Dean, *Efficient estimation of word representations in vector space*, arXiv preprint arXiv:1301.3781 (2013) (cit. on p. 34).

[45] 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)*, Doha, Qatar: Association for Computational Linguistics, 2014 1532, URL: `https://www.aclweb.org/anthology/D14-1162` (cit. on p. 34).

[46] P. Bojanowski, É. Grave, A. Joulin and T. Mikolov, *Enriching Word Vectors with Subword Information*, Transactions of the Association for Computational Linguistics **5** (2017) 135 (cit. on p. 34).

[47] A. G. Nuzzolese, A. L. Gentile, V. Presutti and A. Gangemi, "Conference Linked Data: The ScholarlyData Project", *The Semantic Web - ISWC 2016 - 15th International Semantic Web Conference, Kobe, Japan, October 17-21, 2016, Proceedings, Part II*, vol. 9982, Lecture Notes in Computer Science, 2016 150, URL: `https://doi.org/10.1007/978-3-319-46547-0%5C_16` (cit. on p. 35).

[48] A. Sinha, Z. Shen, Y. Song, H. Ma, D. Eide, B.-J. (Hsu and K. Wang, "An Overview of Microsoft Academic Service (MAS) and Applications", *Proceedings of the 24th International Conference on World Wide Web*, WWW '15 Companion, Florence, Italy: Association for Computing Machinery, 2015 243, ISBN: 9781450334730, URL: `https://doi.org/10.1145/2740908.2742839` (cit. on p. 35).

[49] A. Swartz, *MusicBrainz: A Semantic Web Service*, IEEE Intell. Syst. **17**.1 (2002) 76, URL: `https://doi.org/10.1109/5254.988466` (cit. on p. 35).

[50] R. Navigli and S. P. Ponzetto, "BabelNet: Building a Very Large Multilingual Semantic Network", *ACL 2010, Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, July 11-16, 2010, Uppsala, Sweden*, The Association for Computer Linguistics, 2010 216, URL: `https://www.aclweb.org/anthology/P10-1023/` (cit. on p. 35).

[51] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak and Z. Ives, *Dbpedia: A nucleus for a web of open data*, Springer, 2007 (cit. on pp. 36, 51).

[52] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. N. Mendes, S. Hellmann, M. Morsey, P. Van Kleef, S. Auer et al., *DBpedia–a large-scale, multilingual knowledge base extracted from Wikipedia*, Semantic Web **6**.2 (2015) 167 (cit. on pp. 36, 83).

[53] 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. 36, 51).

[54] 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*, Santa Fe, New Mexico, USA: Association for Computational Linguistics, 2018 2093, URL: `https://www.aclweb.org/anthology/C18-1178` (cit. on p. 36).

[55] 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**.6 (2017) 895 (cit. on p. 36).

[56] A. Bernstein, E. Kaufmann and C. Kaiser, "Querying the semantic web with ginseng: A guided input natural language search engine", *15th Workshop on Information Technologies and Systems, Las Vegas, NV*, Citeseer, 2005 112 (cit. on p. 37).

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

[58] V. Lopez, V. Uren, E. Motta and M. Pasin, *AquaLog: An ontology-driven question answering system for organizational semantic intranets*, Web Semantics: Science, Services and Agents on the World Wide Web **5**.2 (2007) 72 (cit. on p. 37).

[59] V. Lopez, M. Fernández, E. Motta and N. Stieler, *Poweraqua: Supporting users in querying and exploring the semantic web*, Semantic Web **3**.3 (2012) 249 (cit. on p. 37).

[60] D. Damljanovic, M. Agatonovic and H. Cunningham, "FREyA: An interactive way of querying Linked Data using natural language", *The Semantic Web: ESWC 2011 Workshops*, Springer, 2012 125 (cit. on p. 37).

[61] E. Kaufmann, A. Bernstein and L. Fischer, "NLP-Reduce: A "naïve" but Domain-independent Natural Language Interface for Querying Ontologies", ESWC Zurich, 2007 (cit. on p. 37).

[62] J. Lehmann and L. Bühmann, "Autosparql: Let users query your knowledge base", *The Semantic Web: Research and Applications*, Springer, 2011 63 (cit. on p. 37).

[63] 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 p. 37).

[64] 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", *Information Access Evaluation. Multilinguality, Multimodality, and Visualization*, Springer, 2013 321 (cit. on p. 37).

[65] S. Ferré, "squall2sparql: a Translator from Controlled English to Full SPARQL 1.1", *Work. Multilingual Question Answering over Linked Data (QALD-3)*, 2013 (cit. on p. 37).

[66] 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 (cit. on pp. 37, 65).

[67] A. Marginean, "GFMed: Question answering over biomedical linked data with Grammatical Framework", CLEF, 2014 (cit. on p. 37).

[68] A. Ranta, *Grammatical framework: Programming with multilingual grammars*, CSLI Publications, Center for the Study of Language and Information, 2011 (cit. on p. 37).

[69] T. Hamon, N. Grabar, F. Mougin and F. Thiessard, "Description of the POMELO system for the task 2 of QALD-2014", CLEF, 2014 (cit. on p. 38).

[70] 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*, ACM, 2014 313 (cit. on p. 38).

[71] 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 (cit. on p. 38).

[72] C. Unger, C. Forascu, V. Lopez, A.-C. N. Ngomo, E. Cabrio, P. Cimiano and S. Walter, "Question Answering over Linked Data (QALD-5)", *Working Notes for CLEF 2015 Conference*, 2015 (cit. on pp. 38, 64–66).

[73] T. Kuhn, *A Survey and Classification of Controlled Natural Languages*, Comput. Linguistics **40**.1 (2014) 121, URL: `https://doi.org/10.1162/COLI%5C_a%5C_00168` (cit. on p. 38).

[74] D. Diefenbach, A. Both, K. Singh and P. Maret, *Towards a question answering system over the Semantic Web*, Semantic Web **11**.3 (2020) 421, URL: `https://doi.org/10.3233/SW-190343` (cit. on pp. 38, 79).

[75] 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*, CoRR **abs/1907.09361** (2019), arXiv: `1907.09361`, URL: `http://arxiv.org/abs/1907.09361` (cit. on p. 39).

[76] S. Mohammed, P. Shi and J. Lin, "Strong Baselines for Simple Question Answering over Knowledge Graphs with and without Neural Networks", *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 2 (Short Papers)*, Association for Computational Linguistics, 2018 291, URL: `https://doi.org/10.18653/v1/n18-2047` (cit. on p. 39).

[77] 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, WWW 2017, Perth, Australia, April 3-7, 2017*, ACM, 2017 1211, URL: `https://doi.org/10.1145/3038912.3052675` (cit. on pp. 39, 40, 71, 83).

[78] W.-t. Yih, M.-W. Chang, X. He and J. Gao, “Semantic Parsing via Staged Query Graph Generation: Question Answering with Knowledge Base”, *Proceedings of the 53rd ACL Conf.* Vol. 1, 2015 1321 (cit. on pp. 40, 46, 96).

[79] X. He and D. Golub, “Character-Level Question Answering with Attention”, *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, The Association for Computational Linguistics, 2016 1598, URL: `https://doi.org/10.18653/v1/d16-1166` (cit. on p. 40).

[80] D. Guo, D. Tang, N. Duan, M. Zhou and J. Yin, “Dialog-to-Action: Conversational Question Answering Over a Large-Scale Knowledge Base”, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada*, 2018 2946, URL: `http://papers.nips.cc/paper/7558-dialog-to-action-conversational-question-answering-over-a-large-scale-knowledge-base` (cit. on p. 41).

[81] 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 (Volume 1: Long Papers)*, Sofia, Bulgaria: Association for Computational Linguistics, 2013 423, URL: `https://www.aclweb.org/anthology/P13-1042` (cit. on pp. 41, 43, 70–72, 88).

[82] A. Bordes, N. Usunier, S. Chopra and J. Weston, *Large-scale Simple Question Answering with Memory Networks*, 2015, arXiv: `1506.02075 [cs.LG]` (cit. on pp. 41, 115).

[83] A. Talmor and J. Berant, “The Web as a Knowledge-Base for Answering Complex Questions”, *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, 2018 641 (cit. on pp. 43, 88, 115).

[84] R. Collobert and J. Weston, “A unified architecture for natural language processing: deep neural networks with multitask learning”, *Machine Learning, Proceedings of the Twenty-Fifth International Conference (ICML 2008), Helsinki, Finland, June 5-9, 2008*, vol. 307, ACM International Conference Proceeding Series, ACM, 2008 160, URL: `https://doi.org/10.1145/1390156.1390177` (cit. on p. 43).

[85] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu and P. P. Kuksa, *Natural Language Processing (Almost) from Scratch*, J. Mach. Learn. Res. **12** (2011) 2493, URL: `http://dl.acm.org/citation.cfm?id=2078186` (cit. on p. 43).

[86] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado and J. Dean, "Distributed Representations of Words and Phrases and their Compositionality", *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, 2013 3111, URL: `http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality` (cit. on p. 44).

[87] 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. 44, 106).

[88] Y. Yang and M.-W. Chang, "S-MART: Novel Tree-based Structured Learning Algorithms Applied to Tweet Entity Linking.", *ACL 2015*, 2015 (cit. on p. 44).

[89] A. Sakor, I. O. Mulang, K. Singh, S. Shekarpour, M. E. Vidal, J. Lehmann and S. Auer, "Old is gold: linguistic driven approach for entity and relation linking of short text", *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, 2019 2336 (cit. on pp. 44, 80).

[90] X. Lin, H. Li, H. Xin, Z. Li and L. Chen, *KBPearl: A Knowledge Base Population System Supported by Joint Entity and Relation Linking*, Proc. VLDB Endow. **13**.7 (2020) 1035, ISSN: 2150-8097, URL: `https://doi.org/10.14778/3384345.3384352` (cit. on p. 44).

[91] A. Sil and A. Yates, "Re-ranking for joint named-entity recognition and linking", *22nd ACM International Conference on Information and Knowledge Management, CIKM'13, San Francisco, CA, USA, October 27 - November 1, 2013*, ACM, 2013 2369, URL: `https://doi.org/10.1145/2505515.2505601` (cit. on p. 44).

[92] G. Luo, X. Huang, C.-Y. Lin and Z. Nie, "Joint Entity Recognition and Disambiguation", *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, Lisbon, Portugal: Association for Computational Linguistics, 2015 879, URL: `https://www.aclweb.org/anthology/D15-1104` (cit. on p. 44).

[93] D. B. Nguyen, M. Theobald and G. Weikum, *J-NERD: Joint Named Entity Recognition and Disambiguation with Rich Linguistic Features*, Transactions of the Association for Computational Linguistics **4** (2016) 215, URL: `https://www.aclweb.org/anthology/Q16-1016` (cit. on p. 44).

[94] N. Kolitsas, O.-E. Ganea and T. Hofmann, *End-to-End Neural Entity Linking*, Proceedings of the 22nd Conference on Computational Natural Language Learning (2018), URL: `http://dx.doi.org/10.18653/v1/k18-1050` (cit. on p. 44).

[95] D. Vrandečić and M. Krötzsch, *Wikidata: a free collaborative knowledgebase*, Communications of the ACM **57**.10 (2014) 78 (cit. on pp. 44, 83).

[96] A. Delpeuch, *Opentapioca: Lightweight entity linking for wikidata*, arXiv preprint arXiv:1904.09131 (2019) (cit. on p. 44).

[97] A. Sakor, K. Singh, A. Patel and M.-E. Vidal, *FALCON 2.0: An Entity and Relation Linking Tool over Wikidata*, 2019, arXiv: `1912.11270 [cs.CL]` (cit. on p. 44).

[98] Ö. Sevgili, A. Panchenko and C. Biemann,
"Improving Neural Entity Disambiguation with Graph Embeddings", *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, Florence, Italy: Association for Computational Linguistics, 2019 315,
URL: `https://www.aclweb.org/anthology/P19-2044` (cit. on p. 44).

[99] B. Perozzi, R. Al-Rfou and S. Skiena, *DeepWalk*, Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '14 (2014),
URL: `http://dx.doi.org/10.1145/2623330.2623732` (cit. on p. 44).

[100] D. Sorokin and I. Gurevych, "Mixing Context Granularities for Improved Entity Linking on Question Answering Data across Entity Categories",
Proceedings of the Seventh Joint Conference on Lexical and Computational Semantics,
New Orleans, Louisiana: Association for Computational Linguistics, 2018 65,
URL: `https://www.aclweb.org/anthology/S18-2007` (cit. on p. 44).

[101] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston and O. Yakhnenko,
"Translating embeddings for modeling multi-relational data",
Advances in neural information processing systems, 2013 2787 (cit. on p. 44).

[102] 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. 45, 106).

[103] D. Gerber and A. N. Ngomo,
"Extracting Multilingual Natural-Language Patterns for RDF Predicates",
Knowledge Engineering and Knowledge Management - 18th International Conference, EKAW 2012, Galway City, Ireland, October 8-12, 2012. Proceedings, vol. 7603,
Lecture Notes in Computer Science, Springer, 2012 87,
URL: `https://doi.org/10.1007/978-3-642-33876-2%5C_10` (cit. on p. 45).

[104] H. Elsahar, P. Vougiouklis, A. Remaci, C. Gravier, J. Hare, F. Laforest and E. Simperl,
"T-REx: A Large Scale Alignment of Natural Language with Knowledge Base Triples",
Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018), Miyazaki, Japan: European Language Resources Association (ELRA), 2018,
URL: `https://www.aclweb.org/anthology/L18-1544` (cit. on p. 45).

[105] 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 p. 45).

[106] 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 (cit. on pp. 46, 96).

[107] A. N. Ngomo, L. Bühmann, C. Unger, J. Lehmann and D. Gerber,
"SPARQL2NL: verbalizing sparql queries", *22nd International World Wide Web Conference, WWW '13, Rio de Janeiro, Brazil, May 13-17, 2013, Companion Volume*,
International World Wide Web Conferences Steering Committee / ACM, 2013 329,
URL: `https://doi.org/10.1145/2487788.2487936` (cit. on p. 46).

[108] 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*, WWW '13, Rio de Janeiro, Brazil: Association for Computing Machinery, 2013 977, ISBN: 9781450320351, URL: https://doi.org/10.1145/2488388.2488473 (cit. on p. 46).

[109] B. Ell, D. Vrandecic and E. Simperl, "SPARTIQULATION: Verbalizing SPARQL Queries", *The Semantic Web: ESWC 2012 Satellite Events - ESWC 2012 Satellite Events, Heraklion, Crete, Greece, May 27-31, 2012. Revised Selected Papers*, vol. 7540, Lecture Notes in Computer Science, Springer, 2012 117, URL: https://doi.org/10.1007/978-3-662-46641-4%5C_9 (cit. on p. 46).

[110] J. Novikova, O. Dusek and V. Rieser, "The E2E Dataset: New Challenges For End-to-End Generation", *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue, Saarbrücken, Germany, August 15-17, 2017*, Association for Computational Linguistics, 2017 201, URL: https://doi.org/10.18653/v1/w17-5525 (cit. on p. 47).

[111] C. Gardent, A. Shimorina, S. Narayan and L. Perez-Beltrachini, "Creating Training Corpora for NLG Micro-Planners", *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Vancouver, Canada: Association for Computational Linguistics, 2017 179, URL: https://www.aclweb.org/anthology/P17-1017.pdf (cit. on p. 47).

[112] C. Gardent, A. Shimorina, S. Narayan and L. Perez-Beltrachini, "The WebNLG Challenge: Generating Text from RDF Data", *Proceedings of the 10th International Conference on Natural Language Generation*, Santiago de Compostela, Spain: Association for Computational Linguistics, 2017 124, URL: https://www.aclweb.org/anthology/W17-3518 (cit. on p. 48).

[113] S. He, C. Liu, K. Liu and J. Zhao, "Generating Natural Answers by Incorporating Copying and Retrieving Mechanisms in Sequence-to-Sequence Learning", *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Vancouver, Canada: Association for Computational Linguistics, 2017 199, URL: https://www.aclweb.org/anthology/P17-1019 (cit. on pp. 48, 115).

[114] J. Yin, X. Jiang, Z. Lu, L. Shang, H. Li and X. Li, "Neural Generative Question Answering", *Proceedings of the Workshop on Human-Computer Question Answering*, San Diego, California: Association for Computational Linguistics, 2016 36, URL: https://www.aclweb.org/anthology/W16-0106 (cit. on pp. 48, 115).

[115] E. Kacupaj, H. Zafar, M. Maleshkova and J. Lehmann, "VQuAnDa: Verbalization QUestion ANswering DAtaset", *European Semantic Web Conference*, 2020 (cit. on pp. 48, 80, 115, 124).

[116] T. Luong, H. Pham and C. D. Manning, "Effective Approaches to Attention-based Neural Machine Translation", *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, The Association for Computational Linguistics, 2015 1412, URL: https://doi.org/10.18653/v1/d15-1166 (cit. on p. 50).

[117] J. Gehring, M. Auli, D. Grangier, D. Yarats and Y. N. Dauphin, "Convolutional sequence to sequence learning", *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, JMLR. org, 2017 1243 (cit. on pp. 50, 120).

[118] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser and I. Polosukhin, "Attention is all you need", *Advances in neural information processing systems*, 2017 5998 (cit. on pp. 50, 120, 123).

[119] F. M. Suchanek, G. Kasneci and G. Weikum, "Yago: a core of semantic knowledge", *Proceedings of the 16th international conference on World Wide Web*, ACM, 2007 697 (cit. on p. 51).

[120] X. Dong, E. Gabrilovich, G. Heitz, W. Horn, N. Lao, K. Murphy, T. Strohmann, S. Sun and W. Zhang, "Knowledge vault: A web-scale approach to probabilistic knowledge fusion", *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, ACM, 2014 601 (cit. on p. 51).

[121] J. Pérez, M. Arenas and C. Gutierrez, "Semantics and Complexity of SPARQL", *International semantic web conference*, vol. 4273, Springer, 2006 30 (cit. on p. 51).

[122] V. Lopez, C. Unger, P. Cimiano and E. Motta, *Evaluating question answering over linked data*, Web Semantics Science Services And Agents On The World Wide Web **21** (2013) 3, ISSN: 1570-8268 (cit. on p. 52).

[123] 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*, ACM, 2011 1 (cit. on p. 61).

[124] G. A. Miller, *WordNet: a lexical database for English*, Communications of the ACM **38**.11 (1995) 39 (cit. on p. 61).

[125] D. Gerber and A.-C. Ngonga Ngomo, "Bootstrapping the Linked Data Web", *1st Workshop on Web Scale Knowledge Extraction @ ISWC 2011*, 2011 (cit. on p. 61).

[126] G. Maheshwari, M. Dubey, P. Trivedi and J. Lehmann, "How to Revert Question Answering on Knowledge Graphs", *Proceedings of the ISWC 2017 Posters & Demonstrations and Industry Tracks co-located with 16th International Semantic Web Conference (ISWC 2017), Vienna, Austria, October 23rd - to - 25th, 2017*, vol. 1963, CEUR Workshop Proceedings, CEUR-WS.org, 2017, URL: `http://ceur-ws.org/Vol-1963/paper604.pdf` (cit. on p. 70).

[127] C. Unger, A.-C. N. Ngomo and E. Cabrio, "6th open challenge on question answering over linked data (qald-6)", *Semantic Web Evaluation Challenge*, 2016 171 (cit. on pp. 70–72).

[128] Y. Zhang, K. Liu, S. He, G. Ji, Z. Liu, H. Wu and J. Zhao, *Question Answering over Knowledge Base with Neural Attention Combining Global Knowledge Information*, arXiv preprint arXiv:1606.00979 (2016) (cit. on p. 71).

[129] K. Höffner, S. Walter, E. Marx, R. Usbeck, J. Lehmann and A. N. Ngomo, *Survey on challenges of Question Answering in the Semantic Web*, Semantic Web **8**.6 (2017) 895, URL: `https://doi.org/10.3233/SW-160247` (cit. on p. 71).

[130] B. Ell, D. Vrandečić and E. Simperl, “Spartiqulation: Verbalizing sparql queries”, *Extended Semantic Web Conference*, 2012 117 (cit. on p. 71).

[131] 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 World Wide Web conference*, 2013 977 (cit. on p. 71).

[132] S. Vakulenko, J. D. F. Garcia, A. Polleres, M. de Rijke and M. Cochez, “Message Passing for Complex Question Answering over Knowledge Graphs”, *Proceedings of the 28th ACM International Conference on Information and Knowledge Management, CIKM 2019, Beijing, China, November 3-7, 2019*, ACM, 2019 1431, URL: `https://doi.org/10.1145/3357384.3358026` (cit. on p. 79).

[133] H. Zafar, G. Napolitano and J. Lehmann, “Formal query generation for question answering over knowledge bases”, *European Semantic Web Conference*, Springer, 2018 (cit. on pp. 80, 85).

[134] H. Zafar, M. Tavakol and J. Lehmann, *Distantly Supervised Question Parsing*, (2020) (cit. on p. 80).

[135] A. Ismayilov, D. Kontokostas, S. Auer, J. Lehmann, S. Hellmann et al., *Wikidata through the Eyes of DBpedia*, Semantic Web **9**.4 (2018) 493 (cit. on pp. 83, 84).

[136] 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)*, vol. 1, 2014 1415 (cit. on p. 84).

[137] J. Devlin, M. Chang, K. Lee and K. Toutanova, *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*, CoRR **abs/1810.04805** (2018), arXiv: `1810.04805`, URL: `http://arxiv.org/abs/1810.04805` (cit. on p. 87).

[138] D. Banerjee, M. Dubey, D. Chaudhuri and J. Lehmann, “Joint Entity and Relation Linking Using EARL”, *Proceedings of the ISWC 2018 Posters & Demonstrations, Industry and Blue Sky Ideas Tracks co-located with 17th International Semantic Web Conference (ISWC 2018), Monterey, USA, October 8th - to - 12th, 2018*, vol. 2180, CEUR Workshop Proceedings, CEUR-WS.org, 2018, URL: `http://ceur-ws.org/Vol-2180/paper-06.pdf` (cit. on p. 94).

[139] 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. 96).

[140] A. P. B. Veyseh, “Cross-Lingual Question Answering Using Common Semantic Space.”, *TextGraphs@ NAACL-HLT*, 2016 15 (cit. on p. 96).

[141] S. Park, S. Kwon, B. Kim and G. G. Lee, “ISOFT at QALD-5: Hybrid Question Answering System over Linked Data and Text Data.”, *CLEF (Working Notes)*, 2015 (cit. on p. 96).

[142] R. J. Trudeau, *Introduction to Graph Theory (Corrected, enlarged republication. ed.)* 1993 (cit. on p. 96).

[143] 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**.Aug (2011) 2493 (cit. on p. 97).

[144] Y. Pinter, R. Guthrie and J. Eisenstein, “Mimicking Word Embeddings using Subword RNNs”, *EMNLP*, 2017 102 (cit. on p. 97).

[145] P. Ristoski and H. Paulheim, “Rdf2vec: Rdf graph embeddings for data mining”, *International Semantic Web Conference*, Springer, 2016 498 (cit. on p. 97).

[146] D. Kingma and J. Ba, *Adam: A method for stochastic optimization*, arXiv preprint arXiv:1412.6980 (2014) (cit. on p. 97).

[147] B. H. Bloom, *Space/Time Trade-Offs in Hash Coding with Allowable Errors*, Commun. ACM **13**.7 (1970) 422, ISSN: 0001-0782, URL: `https://doi.org/10.1145/362686.362692` (cit. on p. 98).

[148] G. Laporte, H. Mercure and Y. Nobert, *Generalized travelling salesman problem through n sets of nodes: the asymmetrical case*, Discrete Applied Mathematics **18**.2 (1987) 185, ISSN: 0166-218X, URL: `http://www.sciencedirect.com/science/article/pii/0166218X87900205` (cit. on p. 99).

[149] K. Helsgaun, *Solving the equality generalized traveling salesman problem using the Lin–Kernighan–Helsgaun Algorithm*, Mathematical Programming Computation (2015) (cit. on p. 99).

[150] R. Speck and A.-C. Ngonga Ngomo, “Ensemble Learning for Named Entity Recognition”, *The Semantic Web – ISWC 2014*, vol. 8796, Lecture Notes in Computer Science, Springer International Publishing, 2014 519 (cit. on p. 106).

[151] K. Singh, I. O. Mulang, I. Lytra, M. Y. Jaradeh, A. Sakor, M.-E. 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. 106).

[152] E. Colin, C. Gardent, Y. M’rabet, S. Narayan and L. Perez-Beltrachini, “The WebNLG Challenge: Generating Text from DBPedia Data”, *Proceedings of the 9th International Natural Language Generation conference*, Edinburgh, UK: Association for Computational Linguistics, 2016 163, URL: `https://www.aclweb.org/anthology/W16-6626` (cit. on pp. 112, 115).

[153] 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**.2 (2015) 167, URL: `https://doi.org/10.3233/SW-140134` (cit. on p. 112).

[154] D. Diefenbach, Y. Dridi, K. D. Singh and P. Maret, “SPARQLtoUser: Did the Question Answering System Understand me?”, *BLINK/NLIWoD3@ISWC*, 2017 (cit. on p. 113).

[155] P. Trivedi, G. Maheshwari, M. Dubey and J. Lehmann, “LC-QuAD: A Corpus for Complex Question Answering over Knowledge Graphs”, *The Semantic Web - ISWC - 16th International Semantic Web Conference, Vienna, Austria, Proceedings, Part II*, Springer, 2017, URL: `https://doi.org/10.1007/978-3-319-68204-4%5C_22` (cit. on p. 115).

[156] P. Rajpurkar, J. Zhang, K. Lopyrev and P. Liang, "SQuAD: 100,000+ Questions for Machine Comprehension of Text", *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 2016 2383 (cit. on p. 115).

[157] T. Kwiatkowski, J. Palomaki, O. Redfield, M. Collins, A. Parikh, C. Alberti, D. Epstein, I. Polosukhin, M. Kelcey, J. Devlin, K. Lee, K. N. Toutanova, L. Jones, M.-W. Chang, A. Dai, J. Uszkoreit, Q. Le and S. Petrov, *Natural Questions: a Benchmark for Question Answering Research*, Transactions of the Association of Computational Linguistics (2019) (cit. on p. 115).

[158] W.-t. Yih, M. Richardson, C. Meek, M.-W. Chang and J. Suh, "The Value of Semantic Parse Labeling for Knowledge Base Question Answering", *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, Berlin, Germany: Association for Computational Linguistics, 2016 201, URL: `https://www.aclweb.org/anthology/P16-2033` (cit. on p. 115).

[159] E. Choi, H. He, M. Iyyer, M. Yatskar, W.-t. Yih, Y. Choi, P. Liang and L. Zettlemoyer, "QuAC: Question Answering in Context", *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018 2174 (cit. on p. 115).

[160] S. Reddy, D. Chen and C. D. Manning, *Coqa: A conversational question answering challenge*, Transactions of the Association for Computational Linguistics **7** (2019) 249 (cit. on p. 115).

[161] 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*, (2018), eprint: `arXiv:1801.10314` (cit. on pp. 115, 116).

[162] M. Iyyer, W.-t. Yih and M.-W. Chang, "Search-based Neural Structured Learning for Sequential Question Answering", *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Vancouver, Canada: Association for Computational Linguistics, 2017 1821, URL: `https://www.aclweb.org/anthology/P17-1167` (cit. on p. 115).

[163] D. Diefenbach, T. P. Tanon, K. D. Singh and P. Maret, "Question Answering Benchmarks for Wikidata", *Proceedings of the ISWC 2017 Posters & Demonstrations and Industry Tracks co-located with 16th International Semantic Web Conference , Vienna, Austria*, 2017, URL: `http://ceur-ws.org/Vol-1963/paper555.pdf` (cit. on pp. 116, 117).

[164] D. Vrandečić and M. Krötzsch, *Wikidata: A Free Collaborative Knowledge Base*, Communications of the ACM **57** (2014) 78, URL: `http://cacm.acm.org/magazines/2014/10/178785-wikidata/fulltext` (cit. on p. 116).

[165] J. Devlin, M.-W. Chang, K. Lee and K. Toutanova, *Bert: Pre-training of deep bidirectional transformers for language understanding*, arXiv preprint arXiv:1810.04805 (2018) (cit. on p. 120).

[166] A. Radford, K. Narasimhan, T. Salimans and I. Sutskever, *Improving language understanding by generative pre-training*, URL https://s3-us-west-2. amazonaws. com/openai-assets/researchcovers/languageunsupervised/language understanding paper. pdf (2018) (cit. on p. 120).

[167] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer and V. Stoyanov, *Roberta: A robustly optimized bert pretraining approach*, arXiv preprint arXiv:1907.11692 (2019) (cit. on p. 120).

[168] 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 on Association for Computational Linguistics*, ACL, Philadelphia, Pennsylvania: Association for Computational Linguistics, 2002, URL: `https://doi.org/10.3115/1073083.1073135` (cit. on p. 123).

Appendix

APPENDIX A

List of Publications

- *Conference Papers (peer reviewed)*

1. **Mohnish Dubey**, Sourish Dasgupta, Ankit Sharma, Konrad Höffner, Jens Lehmann . "Asknow: A framework for natural language query formalization in sparql." In European Semantic Web Conference, pp. 300-316. Springer, Cham, 2016. DOI: `http://dx.doi.org/10.1007/978-3-319-34129-3_19`
2. Priyansh Trivedi, Gaurav Maheshwari, **Mohnish Dubey**, and Jens Lehmann. "LC-QuAD: A corpus for complex question answering over knowledge graphs." In International Semantic Web Conference, pp. 210-218. Springer, Cham, 2017. The first three authors contributed equally. DOI: `http://dx.doi.org/10.1007/978-3-319-68204-4_22`
3. **Mohnish Dubey**, Debayan Banerjee, Debanjan Chaudhuri, and Jens Lehmann. "Earl: Joint entity and relation linking for question answering over knowledge graphs." In International Semantic Web Conference, pp. 108-126. Springer, Cham, 2018. DOI: `http://dx.doi.org/10.1007/978-3-030-00671-6_7`
4. **Mohnish Dubey**, Debayan Banerjee, Abdelrahman Abdelkawi, and Jens Lehmann. "Lc-quad 2.0 A large dataset for complex question answering over wikidata and dbpedia." In International Semantic Web Conference. Springer. 2019. DOI: `http://dx.doi.org/10.1007/978-3-030-30796-7_5`

- *Demo Papers (peer reviewed)*

5. Gaurav Maheshwari, **Mohnish Dubey**, Priyansh Trivedi, and Jens Lehmann. "How to Revert Question Answering on Knowledge Graphs." In International Semantic Web Conference (Posters, Demos and Industry Tracks). 2017. URL: `http://ceur-ws.org/Vol-1963/paper604.pdf`
6. Debayan Banerjee, **Mohnish Dubey**, Debanjan Chaudhuri, and Jens Lehmann. "Joint Entity and Relation Linking Using EARL." In International Semantic Web Conference (P&D/Industry/BlueSky). 2018. URL: `http://ceur-ws.org/Vol-2180/paper-06.pdf`

List of Figures

List of Tables