universität innsbruck

Question Answering over Knowledge Graphs

Sareh Aghaei Dinani

Universität Innsbruck

Department of Computer Science

submitted to the Faculty of Mathematics, Computer Science and Physics of the University of Innsbruck

in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Computer Science

Primary Supervisor: Prof. Dr. **Anna Fensel**

Secondary Supervisor: Univ.-Prof. Dr. **Adam Jatowt**

Declaration

I, **Sareh Aghaei Dinani**, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the thesis.

Sareh Aghaei Dinani
Innsbruck, Austria
Winter 2023

Abstract

The semantic web contains a large amount of data in the form of knowledge graphs (KGs). KGs, made in Resource Description Framework (RDF) data model, provide a way to represent knowledge from heterogeneous, non-integrated, and inconsistent data. Simple Protocol and RDF Query Language (SPARQL), a query language, is used to build queries and retrieve the information to access the data in KGs. Working with SPARQL to express information needs and explore KGs is difficult for users who are unfamiliar with query languages and the structure of the underlying KGs. Consequently, question answering over KGs has emerged to make them more accessible and seek answers to questions expressed in a natural language manner.

This thesis aims to delve deep into the research of answering natural language questions posed by users across KGs. Question answering over KGs is a challenging task that targets designing systems that are capable of answering users' questions using the RDF facts stored in KGs. This thesis focuses on providing approaches to retrieve answers to factoid questions, either simple or complex.

To assist end-users in accessing KGs created on small and medium scales, a semantic parsing-based approach is proposed, including the offline phase and the semantic parsing phase. The main objective of this approach is to generate training samples in the offline phase and then use the samples to train machine learning models employed in the semantic parsing phase to parse a given input question and automatically find the corresponding SPARQL query. To showcase the proposed approach, a KG describing touristic entities is applied. Furthermore, a graph-driven approach is proposed to bridge the gap between natural language questions and SPARQL queries in huge KGs. Given a question, a graph is built to represent the question's intention as well as a subgraph of the underlying KG is extracted to reduce the huge search space. Then, the question graph is matched over the subgraph to find a query graph corresponding to a SPARQL query, and finally, the query is executed to return the answers. Experiments are conducted to evaluate the approach and exhibit its results.

The achieved results of the proposed approaches indicate the performance in terms of recall, precision, and F1-score and improvements compared to the state-of-the-art. However, there still exists much space for improvement of question answering over KGs.

Acknowledgements

Words cannot express my gratitude to my supervisor, Prof. Dr. Anna Fensel, for her invaluable patience, guidance, and feedback throughout the PhD project. I sincerely thank Univ.-Prof. Dr. Adam Jatowt for accepting to be my second supervisor. I also would like to thank Univ.-Prof. Dr. Dieter Fensel and the whole STI Innsbruck team at the University of Innsbruck.

Additionally, this endeavor would not have been possible without the support from the project WordLiftNG within the Eureka, Eurostars Programme (grant agreement number 877857 with the Austrian Research Promotion Agency (FFG)), and the project KI-NET within the Interreg Osterreich-Bayern 2014-2020 programme (grant agreement number AB 292), that financed my research. I feel very fortunate that this work has furthermore been supported with the 2022 Netidee grant by the Internet Foundation Austria.

I greatly appreciate all the discussions I had with Tek Raj Chhetri, Anelia Kurteva, and Amar Tauqeer. Finally, I am grateful to my friends and colleagues for their moral support.

Dedication

To my family
To my friends
For their endless love, support, and encouragement

Contents

Chapter 1

Introduction

Contents

1.1 Overview

Knowledge graphs (KGs) are an emerging semi-structured form of knowledge representation promoted by the semantic web communities to present real-world entities, properties, and relations between these entities as graphs. The semantic web has supported the emergence of an enormous amount of information in the form of KGs. KGs, represented in Resource Description Framework (RDF), consist of RDF triples (or facts[1]) in the form of subject-predicate-object.

Question answering (QA) over KGs is a developing research area [1, 2, 3] that brings together research from semantic web technology, machine learning (ML), and natural language processing (NLP) to create and develop systems that make KGs accessible to users. Simple Protocol and RDF Query Language (SPARQL) is a flexible query language that allows writing queries for retrieving information over KGs. However, working with SPARQL is difficult or even impossible for end-users who are not familiar with query languages and data structures, reinforcing the importance of QA over KGs. To provide KG-based question answering systems (KGQASs), there are mainly three research directions. The first is to rely on rules to answer questions [4, 5, 6]. The second is information retrieval-based KGQASs that first retrieve all candidate answers and then choose the most appropriate answers [7, 8, 9]. Finally, there is a possibility to parse natural language questions (NLQs) and convert them to executable queries [10, 11, 12].

The focus of this thesis is to design and build semantic parsing-based KGQASs to automatically search a collection of triples and find the answer to a question. For example, in the healthcare domain, given the question "Which cancers share at least one protein with breast cancer?", the answers are retrieved from the triples containing the information.

[1]The terms triple and fact are used interchangeably in this research.

1.2 Motivation

Today, people are surrounded by a massive amount of data, so they prefer to get an exact and precise answer to a question in a natural way rather than exploring and identifying it from such a large amount of data [13]. Not only do people incline to get straight answers to their questions in daily life (e.g., "where is the nearest bus station?"), but also industrial scenarios such as manufacturing and health explicitly deal with QA in their processes (e.g., "How many failures associated with scrapped boards in a given time frame?", "What are the symptoms of COVID-19?") [13]. These examples demonstrate how QA systems (QASs) have a significant influence in everyday life, as well as industry [14]. It is no wonder that there is no traditional web search in the future vision, and QASs effectively create a strong brand image.

Recent advancements such as the development of language models like OpenAI's ChatGPT[2] or virtual assistants like Siri[3], are capable of answering questions in a conversational manner. In light of these trends, the development of effective QASs over KGs is a crucial area of research, especially when it comes to handling complex queries that require a deep understanding of the underlying knowledge. This is where KGs come into play, as they provide a structured representation of knowledge that can be leveraged to enhance the accuracy and efficiency of QASs.

In the context of the data web and standards like RDF, KGs are being extensively developed and enriched to represent, model, publish, and store data in a semantically structured way [15]. In addition to the surge of adopting KGs in large and well-known brands and companies (such as NASA[4], and Netflix[5]), medium-sized enterprises (SMEs) (such

[2] https://openai.com/blog/chatgpt/
[3] https://www.apple.com/es/siri/
[4] https://www.nasa.gov/
[5] https://www.netflix.com

as Onlim[6]) are now harnessing the power of KGs to revolutionize their business, revealing hidden insights and accelerating product development cycles [16].

Furthermore, research projects like WordLiftNG [17], and KI-Net [18], that this work is a part of, add explicit motivation use cases and scenarios for applying KGs within the market and unlocking hidden knowledge to users through QA over KGs. The WorLiftNG project aims to construct SEO-friendly websites using KGs for SMEs (e.g., in the domain of tourism) and the KI-Net project aims to build intelligent blocks for optimizing industrial manufacturing processes (e.g., using KGs for representing data generated by different machines during manufacturing processes and then answering the arising questions relating to the production process). Another use case that constitutes an additional motivation is to make health information on smell and taste disorders accessible to users by creating a KG with real data from Smell and Taste Center in Hospital Gelderse Vallei[7].

Exploring and querying KGs representing heterogeneous and inconsistent data is an elusive and long-standing challenge [19]. Despite the advancement in QA over KGs, an immense effort still is required to develop and improve KGQASs and facilitate access to knowledge stored in KGs [20]. The questions that humans tend to are straightforward and factoid questions beginning with WH words, such as what, where, who, whose, when, which, and how many (e.g., "When does Mozart Week begin?", "What ski resort has the most slopes in Austria?") [21]. Overall, the exponential growth of KGs in various domains and meeting users' information needs by obtaining direct answers to their factoid questions, motivate the research interest in designing and developing solutions to answer questions asked by users.

[6] `https://onlinecompany.com/`
[7] `https://www.geldersevallei.nl/home`

1.3 Challenges and Research Questions

This section introduces the major challenges addressed in this thesis and then formulates the research questions.

The primary challenges that are addressed in this thesis to design and build semantic parsing-based KGQASs are summarized as follows:

- Search Space: Large KGs such as DBPedia [22] with millions or billions of facts are ideal sources for answering factoid questions [23, 24]. The presence of this large number of facts involves millions of entities and thousands of relations. Considering the whole such large KGs for each question is an intricate challenge in KGQASs. For efficiency, these systems require pruning irrelevant parts of the underlying KG and reducing search space by extracting subsets that capture answers with respect to input questions.

- Training Examples: Despite numerous possibilities to use ML techniques to develop KGQASs, training data is a critical challenge since training examples are hardly available in real settings. For example, there is often little or no training data to develop QASs across small and medium-scaled KGs that are mainly created in SMEs.

- Lexical Gaps: Since questions are posed by end-users based on their own vocabularies, questions can be expressed differently. Bridging the lexical gaps and ambiguities between questions and KGs' vocabularies is another challenge in KGQASs. For example, the relation "leader" can be expressed in multiple ways such as "manager", "head", or "master".

In light of the motivation and described challenges, the underlying hypothesis of this thesis is as follows: ***"Semantic web technologies, ML, and NLP can improve the performance of QA over KGs."***

The formulated hypothesis determines the research questions that are the thesis's basis. This work intends to seek the research questions to evaluate the hypothesis and then prove or reject it. The research questions are summarized as follows:

RQ1
How to bridge the gap between NLQs (i.e., unstructured) and SPARQL queries (i.e., structured) in semantic parsing-based KGQASs?

RQ2
How to extract a subset of a huge KG for a posed NLQ that reduces the size of the KG while covering the answer?

RQ3
How to alleviate the problem of insufficient amount of training data in QA over KGs in SMEs?

RQ4
How to overcome ambiguities in questions' phrases and KG's vocabularies?

1.4 Research Methodology

This section presents an overview of this thesis's research methodology. The research methodology consists of the following steps:

- Literature Review: The initial step is to understand the problem, recognize what has been concluded in this research area, and identify research gaps. The literature review includes the reviews of the scientific papers, following the scope of the work, using semantic web, ML and NLP technologies. This investigation to explore

different research conducted to provide QA over KGs, helps to find the research gaps. The result of this exploration is represented in Chapter 2. Moreover, to collect the research requirements, the documents describing the requirements of the WordLiftNG project and the KI-Net project are examined.

- Design and Implementation: After performing a literature review and understanding the project requirements, the next step is to design an approach and perform the implementation. The case studies are undertaken inline with the projects' requirements and identified research gaps. To fulfill RQ1, according to the outcomes of the literature review, a need to design and develop a semantic parser to convert NLQs to SPARQL queries is recognized. Therefore, a solution leveraging NLP and ML techniques concerning the size of the KGs is modeled. To identify answer search space reduction in huge KGs (RQ2), Personal Page Rank (PPR) [25] is adopted to design a new approach for extracting a part of the KG as a knowledge subgraph that is likely to contain the answer to a particular input question. Moreover, a solution is developed to generate training examples and accomplish RQ3 according to the literature review and the requirements of the WordLiftNG project. Meanwhile, NLP techniques such as word embedding and semantic similarity are applied to overcome the ambiguities between NLQs and KGs and output RQ4.

- Evaluation: The designed and developed approaches for the conducted case studies are derived from the projects' requirements and standard evaluation metrics, including recall, precision, and F1-score. Moreover, the achieved experimental results are compared to the state-of-the-art. Details of the evaluation are discussed in Section 3.6, Section 4.5, and Section 5.5

1.5 Contributions and Discussion

This section discusses the core contributions of this thesis based on the described research questions in Section 1.3.

- This thesis contains two contributions to bridging the gap between NLQs and SPARQL queries (RQ1). The first contribution is defining query patterns based on graph isomorphisms and automatically mapping questions into the query patterns according to the questions' grammatical information. More details are given in Section 3.5.3. The second contribution is to build a semantic parser in order to convert NLQs into graphs as intermediate forms. More information can be found in Section 5.4.2.

- The research on how to narrow down search space in KGs (RQ2) led to introducing of a generic approach to extract knowledge subgraphs for a given input NLG which greatly decreases the size of the KG while covering the answers. A detailed description of the proposed approach is given in Chapter 4.

- The work on how to overcome the training data problem in SMEs triggered a semi-automatic generic approach to creating training data using RDF triples stored in KGs. The approach is discussed in Section 3.5.2.

- This thesis presents two contributions to tackle ambiguities between question sentences and KGs' vocabularies (RQ4). In the first contribution, different similarity measurements and embedding words are applied, discussed in Section 3.5.3.2 and Section 5.4.3. The second contribution is to take advantage of the probability distributions described in Section 3.5.3.3.

1.6 Structure of the Thesis

This section outlines the organization of this cumulative thesis and presents the publications that contributed to the thesis. It consists of the following chapters:

- Chapter 2: A study has been conducted to get an overview of possible means to fetch relevant content over different information resources on the web that is found in Chapter 2. This study presents the state-of-the-art in the area of QA over KGs. This chapter corresponds to the publication "Interactive Search on the Web: The Story So Far" published in Information journal.

- Chapter 3: A new approach to developing KGQASs over small and medium-scaled KGs is presented[8]. The corresponding publication is "Question Answering over Knowledge Graphs: A Case Study in Tourism" published in IEEE Access journal.

- Chapter 4: It contains contributions on an approach for extracting a small subset from a big KG as a knowledge subgraph that is relevant to a given NLQ[9]. The corresponding publication is "Building Knowledge Subgraphs in Question Answering over Knowledge Graphs" in the 22nd International Conference on Web Engineering.

- Chapter 5: A data-driven approach for QA over huge KGs is described in this Chapter[10]. The corresponding publication is "Question Answering over Knowledge Graphs: A Graph-Driven Approach" in the 21st IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology.

[8]The code is available at `https://github.com/sareaghaei/SMEKGQA`
[9]The code is available at `https://github.com/sareaghaei/SBKGEX`
[10]The code is available at `https://github.com/sareaghaei/GDQA`

1.7 Preliminaries

This section introduces the core concepts and technologies from semantic web (Section 1.7.1), ML (Section 1.7.2) and NLP (Section 1.7.3) used in this thesis as well as an overview of QASs (Section 1.7.4).

1.7.1 Semantic Web

The semantic web, an extended version of the current web, brings structure and meaning to the information described in web pages [26]. The semantic web includes a set of standards and technologies (such as ontologies, data interchange formats, and query languages) to represent the information in a machine-processable structure. In the following, a brief introduction to semantic web technologies [27] is presented, and then KGs powered by semantic web technologies are explained.

- Semantic Web Technology: Semantic web technologies provide solutions for integrating and making sense of heterogeneous data [28]. Here, RDF and SPARQL are described as part of the standards in the semantic web technologies that this thesis relies on.

  RDF is a standard model to describe resources with statements. A statement (or triple or fact) comprises three parts, the subject, the predicate, and the object [29, 30]. The subjects and predicates are Uniform Resource Identifiers (URIs), and the objects are either URIs or literals.

  SPARQL is a structured query language proposed by W3C to query RDF data through following a triple- or graph pattern. A triple pattern is an RDF triple in which zero or more variables might appear. SPARQL holds similarities (e.g., projection, join, union, constraint) with the role of Structured Query Language (SQL) as it pertains to relational databases [20].

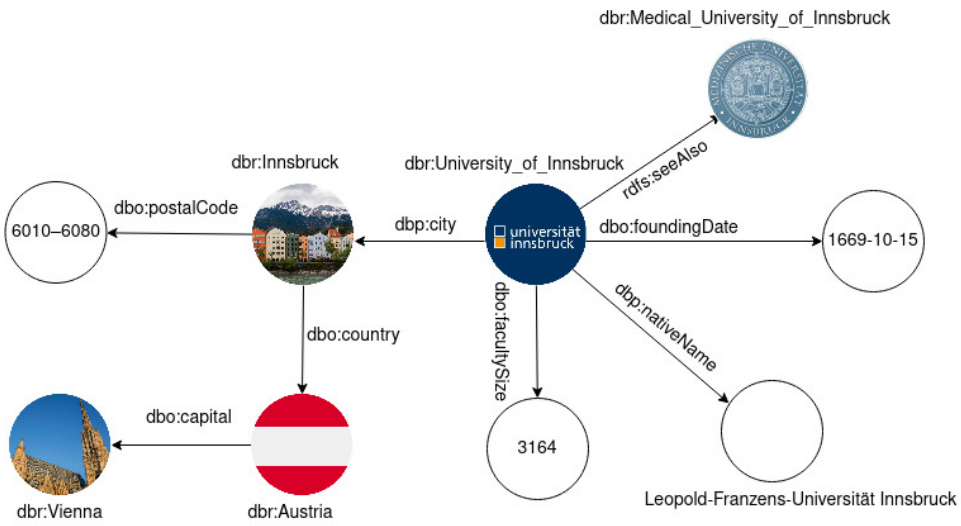
- KGs: The term KG was coined by Google in 2012 as a backbone of a new web search strategy [31]. A KG is a graph-structured model that stores factual information in the form of semantically-interrelated relationships between real-world entities. In contrast to relational databases, this information is semi-structured as a graph rather than structured [32]. Graph structures for representing knowledge have a long history in the fields of logic and artificial intelligence (AI) and are known as semantic networks.

  The representation of the information in the KG using RDF is viewed as RDF KG promoted by the semantic web from the early days [33]. A formal definition of KG is as follows [32]:

  A KG $G = (E, R, L, C)$ is a tuple of pair-wise disjoint sets E, R, L, and C correspond to the set of entities, relations, literals, and types or classes, respectively [34]. A statement in G is modelled as a triple (h, r, t), with $h \subset E \cup R \cup C$, $r \subset R$, and $t \subset E \cup R \cup L \cup C$ [34].

  In the above definition, an entity $e \subset E$ is a real-world or abstract object determined by a URI. A relation $r \subset R$ describes entities identified by a URI. A literal $l \subset L$ represents value of a datatype (e.g., date, string, integer). A Class $c \subset C$ is a set of logically-relevant entities with similar characteristics.

  Thus, the insertion of the data instances into ontological terms becomes an RDF KG in which entities (i.e., subjects and objects) are the nodes of the graph and relations (i.e., predicates) are the edges of the graph [35]. In an RDF KG, the subject and the predicate are identified by URIs, while the object includes a URI or a literal.



Simple Question - When was the University of Innsbruck founded?

Complex Question - What is the capital of the country where Innsbruck University is located?

Figure 1.1: Examples of simple and complex questions.

In the past years, several publicly available KGs have been developed, such as DBpedia, Wikidata [36], YAGO [37], Google's Knowledge Graph [38], or NELL [39]. Despite the generic and open-world KGs, KGs are growingly adopted for a wide range of settings such as tourism [40], news and media [41], healthcare [42], and manufacturing [43].

Figure 1.1 illustrates a snippet of DBPedia in which knowledge about the University of Innsbruck is encoded (note that the prefixes "dbr", "dbp", "dbo", and "rdfs" refer to the namespace URIs, e.g., "dbr" denotes `http://dbpedia.org/resource/`).

1.7.2 Machine Learning

ML, a discipline of AI, targets to imitate the way that humans learn to develop machines [44]. Thus, machines find the ability to learn from data and past experiences while discovering hidden patterns to make predictions with minimal human intervention [45].

A formal definition of ML, widely quoted, is "a computer program is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T, as measured by P, improves with experience E" [46]. In this definition, the experience E refers to data, making ML a data-driven, black-box, or empirical method [47]. The task T represents the application field of ML models (e.g., a classification task). The performance measure P measures how well the task is performed, namely how accurate the model fits the data and can predict [47].

Based on the given problem and the available data, there are three types of ML, supervised, unsupervised, and reinforcement learning [47, 48]. Supervised learning deals with input data and labels, such as classification tasks [49]. In contrast, unsupervised learning does not have labels of data values and reveals a hidden pattern from the data cluster [49]. Reinforcement learning determines the best actions to maximize rewards earned by agents while achieving a specified goal [50].

The family of artificial neural networks [51] is of significance because its flexible structure allows it to be adjusted for a wide range of settings across all three types of ML [52, 53]. A neural network, inspired by way of information processing in biological nervous systems, includes mathematical representations of connected processing units called artificial neurons [52]. The neurons are organized into networks with different layers (i.e., input layers, hidden layers, and output layers). The number of layers and neurons (i.e., hyperparameters) needs to be set manually or identified by an optimization technique. Deep learning [54] uses deep neural networks (more than one hidden layer) fed with data to automatically discover a representation needed for the given learning task.

1.7.3 Natural Language Processing

NLP, the intersection of AI and linguistics, began in the 1950s [55]. NLP deals with analyzing human communication and enabling machines to work with text. NLP is defined as "Natural language processing employs computational techniques for the purpose of learning, understanding, and producing human language content" [56].

In this thesis, entity linking (EL), syntactic analysis, and word embeddings, as part of the main NLP paradigms, are used and described as follows.

- EL: This, known as record linkage or entity resolution, empowers NLP applications by linking each relevant named entity mention found in a text to an appropriate entry in a knowledge base (e.g., a KG) [57]. The mentions are recognized by the task of Named Entity Recognition (NER). NER deals with identifying and tagging the mentions with their respective types (e.g., person, location). EL can be viewed as the disambiguation of the mentions [58].

- Syntactic Analysis: This involves the analysis of the words in a sentence by following the sentence's grammatical structure [59]. The syntactic analysis pipelines perform tokenization, Part Of Speech (POS) tagging, lemmatization, and dependency parsing [59]. Tokenization is the task of breaking and separating a piece of text into smaller units (e.g., words) called tokens. POS tagging refers to marking a word in a text to a corresponding part of a speech tag, depending on the definition and context. Lemmatization is a process of determining a base or dictionary form (lemma) for a word depending on its meaning. Dependency parsing is a process of examining the dependencies between the words of a sentence in order to determine its grammatical structure.

- Word Embedding: This, an integral part of NLP, is the task of representing words in a shared vector space. Word embedding techniques learn a numerical representation for words that are categorized into non-contextual and contextual techniques. Non-contextual models (e.g., Word2Vec [60], GloVe [61]) generate a single representation for a word that could be used in different sentences with different meanings. Contextual models (e.g., BERT [62]) take into consideration the contexts of words and learn contextual embeddings for the words.

1.7.4 Question Answering Systems

QASs have a pretty long history in computer science starting from late 1960 with the intention of satisfying the need to query information content available in various formats [63, 64]. The first QAS, namely Baseball [65], was developed to access data over databases (i.e., structured data) [64], followed by QASs over text (i.e., unstructured data) referred as information retrieval (IR)-based QASs. IR-based QASs find a small portion of text that contains the answer to a question from a collection of documents. In the last two decades, QA over KGs (i.e., semi-structured) has gained attention with the semantic web's emergence.

QA over KGs is the task of extracting answers in response to NLQs across the KGs. NLQs are categorized into factoid, hypothetical, causal, and confirmation questions [66].

- Factoid questions: These questions are factual, starting with WH words (e.g., who, where), and expecting answers are mostly entities. For example, "What is the phone number of the ski-resorts that open in October?"

- Hypothetical questions: These questions are based on suppositions and have no specific answers. For example, "What would happen

if governments fail to police the tourism industry properly?"

- Causal questions: These questions seek explanations, reasons, or elaborations for certain occurrences or things. For example, "How can tourism have a negative impact on the economy?"

- Confirmation questions: These questions are answered in a yes or no form through verification and justification. For example, "Is Mozart Week a classical music festival in winter?"

Factoid questions, the most asked type of NLQs, are classified into simple questions and complex questions based on the number of KG triples required for reasoning to provide answers.

- A simple question, called single-hop question, is a factoid question requiring one-hop reasoning over facts stored in the KG. For example, "When was the University of Innsbruck founded?" is answered through one fact[11] over the KG presented in Figure 1.1.

- A complex question, namely multi-hop question, is a factoid question requiring two or more facts of the KG. For example, "What country is the university of Innsbruck located in?" requires 2-hop[12] reasoning as illustrated in Figure 1.1.

1.8 Conclusion and Future Work

The main objective of this research is to propose and develop KGQASs to make accessible the knowledge stored in KGs. To achieve the research objective and overcome the research challenges (discussed in Section 1.3), the research problem is broken down into four research questions according to the defined hypothesis. The validation of the hypothesis requires

[11](University_of_Innsbruck, foundingDate, 1669-10-15)
[12](University_of_Innsbruck, city, Innsbruck), (Innsbruck, country, Austria)

answering the research questions as mention in Section 1.3. A summary of answers toward the research questions is given as follows.

RQ1- *How to bridge the gap between NLQs (i.e., unstructured) and SPARQL queries (i.e., structured) in semantic parsing-based KGQASs?*

This research question is addressed by training a classifier to automatically map a given NLQ to a corresponding query pattern in QA over small or medium-scaled KGs. The query patterns are established based on isomorphic graphs and structural equivalence of SPARQL queries. Furthermore, in the proposed approach to developing QA over huge KGs, an intermediate form, namely question graph, is introduced to represent the intention of an input NLQ. The grammatical structure of NLQs is taken into account to transform NLQs to question graphs.

RQ2- *How to extract a subset of a huge KG for a posed NLQ that reduces the size of the KG while covering the answer?*

Given an NLQ, extracting a small subset from the KG reduces the search space and makes the final answer extraction process more straightforward. To obtain subsets and address the *RD2*, a new approach is proposed, including three major stages: topic entity identification, neighborhood retrieval, and knowledge subgraph retrieval. The proposed approach maps entity mentions appearing in a question sentence with their corresponding entities (called topic entities) in the KG and then forms a neighborhood graph by retrieving entities and edges with a maximum distance. To find the more relevant facts from the neighborhood graph, a rank score is calculated for each node through a bi-directed propagation process based on PPR.

RQ3- *How to alleviate the problem of insufficient amount of training data in QA over KGs in SMEs?*

A semi-automatic approach is designed generating questions with their answers (i.e., the training data) over RDF triples to address this

research question. The main reasons for proposing a semi-automatic solution include (1) generating questions manually would be too costly and require a certain level of knowledge of the underlying KG's domain, and (2) automatically generating questions using neural networks or natural language models requires large KGs to achieve good performance that are rarely available in SMEs. In the proposed approach, a set of RDF templates is defined based on the maximum number of hops in complex questions. Then, these RDF templates are utilized to generate NLQs through querying the KG and verbalizing query results.

RQ4- *How to overcome ambiguities in questions' phrases and KG's vocabularies?*

A way is introduced to apply several techniques such as n-gram, string similarity metrics, and word embeddings to tackle the task of entity linking. In addition, a classifier is trained to predict the probability distribution of the relations. Thus, the way to perform entity linking as well as training the classifier to extract relations address this research question for the proposed approach in QA over small and medium-scaled KGs. Furthermore, *RQ4* is answered by exploring the idea of employing structural similarity, string similarity, word embeddings, and semantic networks in the introduced solution for developing QASs over huge KGs.

The answers to the research questions conclude that the hypothesis is successfully validated and indicate that the semantic web technologies, ML, and NLP can improve the performance of QA over KGs.

The list of the publications resulting from this research is available on page 170 as well as the other related scientific publications completed during the PhD term.

Although this research discusses several contributions and has the potential to make a significant impact, there are some limitations (as described in Section Section 3.7 and Section 5.5.3) that should be ac-

knowledged. To develop QASs across small and medium-scaled KGs, the proposed semi-automatic approach for generating questions excludes questions with low frequency. The exclusion of the rare questions induces misclassifications and consequently limited performance. Moreover, the slot filling and query execution step lacks ranking techniques and considers all the permutations. While incorrect permutations negatively affect performance due to the increase in false positives. Also, in QA over huge KGs, the assumption of the availability of at least one topic mention in an input question limits the work as some questions may be no topic mention. Moreover, further investigation is required to define the transformations rules and generate question graphs and the work should be tested on more datasets. While the experimental results (either QA over KGs in SMEs or QA over huge KGs) are positive, the effectiveness and efficiency of approaches have not been tested in real settings. The potential future work can address the mentioned limitations and exploit this research in a real-world application to evaluate its efficiency with regard to different criteria such as time response.

This research direction is still continuing, and several issues and challenges need to be tackled by further research. In the following, future directions are discussed.

- Scalability is a key challenge in KGQASs and its lack leads to the performance issue, especially in terms of response time [67]. For example, as the size of the KGs increases, the time to perform tasks such as search also increases. It would be significant to further develop some techniques, such as parallelization and indexing, to design scalable KGQASs. A similar problem has also been noted by other studies such as Diefenbach et al. [14] or Hu et al. [67].

- A robust KGQAS is able to answer various types of questions (e.g., yes/no questions or even syntactically incorrect factoid questions),

while this thesis focuses on well-formulated factoid questions and may fall short of the other questions. Therefore, answering different types of questions can be considered as further future research direction to address the robustness.

- Multilinguality is an open challenge in QA over KGs. A multilingual KGQAS refers to answering questions in a different language from the underlying KG or answering questions with the same language as the KG but supporting different languages (e.g., German and English).

Chapter 2

Interactive Search on the Web: The Story So Far

Contents

2.1 Abstract

Search on the web, specifically fetching of the relevant content, has been paid attention to since the advent of the web and particularly in recent years due to the tremendous growth in the volume of data and web pages. This paper categorizes the search services from the early days of the web to the present into keyword search engines, semantic search engines, question answering systems, dialogue systems, and chatbots. As the first generation of search engines, keyword search engines have adopted keyword-based techniques to find the web pages containing the query keywords and ranking search results. In contrast, semantic search engines try to find meaningful and accurate results on the meaning and relations of things. Question-answering systems aim to find precise answers to natural language questions rather than returning a ranked list of relevant sources. As a subset of question answering systems, dialogue systems target to interact with human users through a dialog expressed in natural language. As a subset of dialogue systems, chatbots try to simulate human-like conversations. The paper provides an overview of the typical aspects of the studied search services, including process models, data preparation and presentation, common methodologies, and categories.

Keywords: Search on the Web, Keyword Search Engine, Semantic Search Engine, Question Answer System, Dialogue System, Chatbot, Semantic Web, Natural Language Processing, Review

Publication: Sareh Aghaei, Kevin Angele, Elwin Huaman, Geni Bushati, Mathias Schiestl, and Anna Fensel. “Interactive Search on the Web: The Story So Far”. In: Information Vol. 13, No. 7 (2022), p. 324. DOI: 10.1109/ACCESS.2022.3187178

2.2 Introduction

The immense amount of data on the web has created the need for search services to explore and fetch the desired information. With the widespread use of the internet of things and social networks, it is expected to have a further substantial increase in the amount of data generated [68, 69]. Although estimation of the actual amount of data available on the web is difficult, there are some astounding numbers. In 2020, 1.7 MB (Megabytes) of data has been created every second by every person and 463 EB (Exabytes) of data will be generated each day by humans as of 2025[1]. So, with the huge amount of data being generated, it is irrefutable that finding the needed information on the web requires search services which embody extremely powerful and valuable tools for fetching any sort of information from the web [70].

A vast number of techniques and approaches have been proposed to provide search services on the web since the advent of the web [71, 72]. Search services can be essentially viewed as answering machines. Search engines such as Google, Baidu, and Yahoo, with the most traffic [72], scour billions of pieces of the web and evaluate thousands of factors to determine which content is most likely to be relevant [73]. Here, keyword search engines, semantic search engines, question answering systems (QASs), dialogue systems, and chatbots can be considered as the main proposed types of search services.

To the best of our knowledge, there is no literature review dedicated to categorizing and discussing all search services on the web as well as the ways in which they are interrelated. Existing research focuses on either one type of search services (such as [74, 75, 76]) or two types of services (such as [70, 77]). The various presented studies (Sections

[1] `https://www.weforum.org/agenda/2019/04/how-much-data-is-generated-each-day-cf4bddf29f/`

2.4.1, 2.5.1, 2.6.1, 2.7.1 and 2.8.1) mostly conduct a detailed review of one aspect (such as mainstream techniques or categories) while our work investigates different directions (i.e., process models, data preparation and presentation, common methodologies and categories). Moreover, we offer a complete picture of the timeline of the evolution of search services which is missing in the literature.

Keyword search engines, as the first generation of search engines, are keyword-based and have been widely used under the hyperlink data environment [70]. They have adopted the keyword-based technique to find the web pages containing the query keywords and ranking search results [78]. Therefore, it is not easy to get an accurate result as they do not know the exact meaning of the keywords used [79].

Semantic search engines are meaning-based and have become known since the web 3.0 [80]. They make efforts to find meaningful and accurate results upon the meaning and relations of the words [66, 81]. Classification of semantic search engines over user interaction mode include keyword-based semantic service, form-based semantic service, view-based semantic service, and natural language-based semantic service [82].

QASs can give users precise answers to the questions presented in natural language [83]. Given a user's natural language question, the system converts the question to a query and then submit the query to the search engine. Afterward, the system extracts all relevant answers from the search results and finally selects the most desired answers to return [84]. QASs fall into different groups based on the type of answer, including sentence/paragraph-answer based QAS, yes/no-answer based QAS, multimedia-answer based QAS, opinionated-answer based QAS, dialogue-answer based QAS [66].

Dialogue systems (also referred to as conversational systems) are a subset of QASs targeting to interact with humans through a dialog ex-

pressed in natural language. The process model of a dialogue system is composed of three layers: user experience, conversation engine, and data layer. The conversation engine layer, as the middle layer of the model, include natural language understanding, dialogue manager, and natural language generation [85]. The dialogue systems can be divided into two groups, namely, task-oriented and non-task-oriented systems, from the aspect of having a task as the goal or not in the dialogue [86]. So the task-oriented dialogue systems target assisting users in completing tasks, while non-task-oriented dialogue systems such as chatbots aim to make conversation with human beings.

Chatbots are designed to simulate human-like conversations [76]. Chatbots' early and primary role has been constructing extended conversations to mimic chats characteristic of human interaction. However, chatbots have been employed to do practical tasks over time [77, 87]. Chatbots can be divided into three groups upon goals being achieved: informative chatbots, chat-based chatbots, and task-based chatbots [77].

The interaction of the described search services including keyword search engines, semantic search engines, QASs, dialogue systems and chatbots is depicted in Figure 2.1 (e.g., QASs can overlap in some aspects with semantic search engines or dialogue systems are a subclass of QASs).

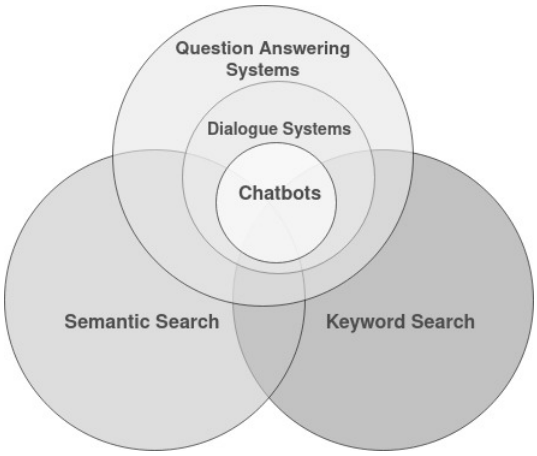


Figure 2.1: The interaction of different types of the search services.

This paper presents the evolution and typical aspects of each search service such as process models, data preparation and presentation, common methodologies, categories (depicted in Figure 2.2) and clarifies their interactions. For each search service, numerous research works have been introduced and developed since the advent of the web which can not be described exhaustively in one paper. Therefore, the most typical development trends are reviewed over time in this paper.

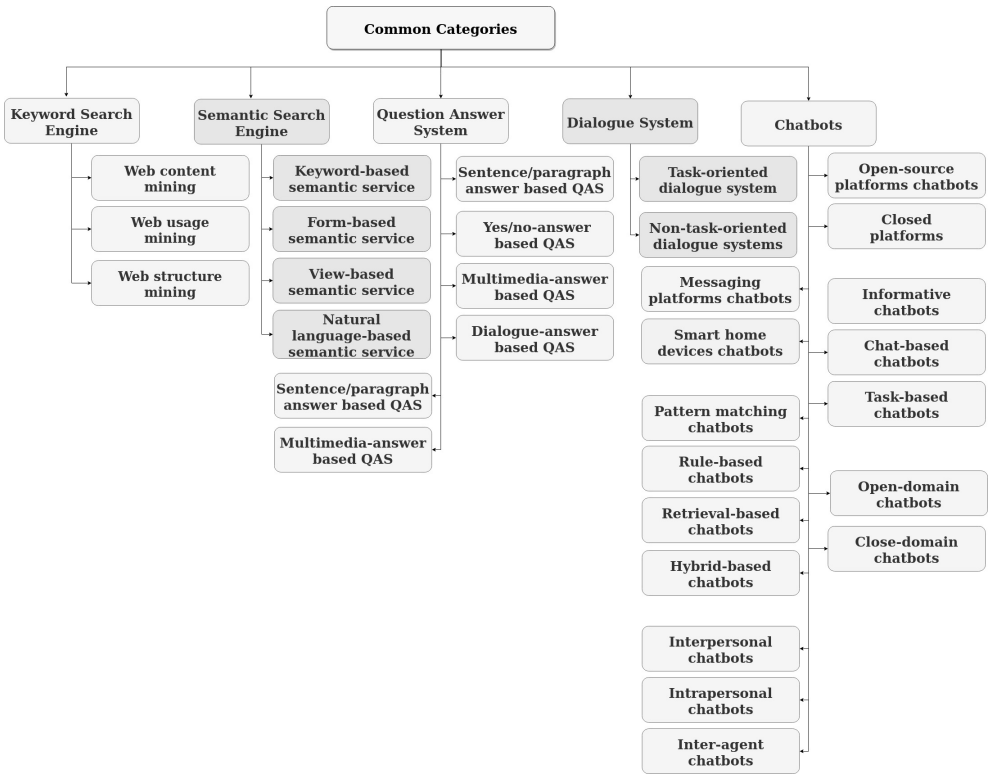


Figure 2.2: The common categories of different types of the search services.

The remainder of the paper is organized as follows. The survey methodology is given in the next section. In Section 2.4, keyword search engines are described. Then, semantic search engines are discussed in Section 2.5. Section 2.6 describes QASs. Dialogue systems and chatbots are explained in Sections 2.7 and 2.8, respectively. The future of search on the web is discussed in Section 2.9. Section 2.10 describes the major findings of the research study. Finally, Section 2.11 concludes the paper.

2.3 Survey Methodology

For this paper, a typical methodology for doing a survey has been followed. The peer-reviewed publications as the primary resource have been identified using the scholarly indexing services: Google Scholar, IEEE (Institute of Electrical and Electronics Engineers) Xplore, ACM (Association of Computing Machinery) Digital Library, Scopus, and DBLP (Digital Bibliography & Library Project). The query terms included keywords "search engines", "semantic web", "semantic search engines", "knowledge graphs", "question-answering systems", "dialogue systems", "chatbots". The selected time frame to find relevant resources was mainly from 2000 to the present, as only keyword search engines were extensively used before. Authors and affiliations of the publications were also used as keywords to find additional relevant sources of knowledge. Moreover, the citation count of the papers has been a key criterion to select and review those. Based on the citation count, for each subsection, the most known papers were examined and reviewed.

In order to provide a comprehensive investigation for each type of search service on the web, the search services, including keyword search engines, semantic search engines, QASs, dialogue systems, and chatbots, are reviewed in separate sections. Each section consists of six subsections: current state of the art and related works, process model, data preparation and representation, common methodologies, categories, and summary.

Firstly, a short history and definition of search services is given and its interaction with the other types of search services is shown. Then the current state of the art and the related works are described in the following subsection. The process model subsection presents a high level conceptual model which defines the structure and behavior of the search service. Next, data preparation and representation are described. The

primary methodologies to develop the search service are expressed in the common methodologies subsection. In the categories subsection, the major groups of the search service are indicated, and finally, a summary of the search service is given in the summary subsection.

2.4 Keyword Search Engine

In the early years of the web, documents were indexed manually by making a list of links hosted on specific web servers. For example, Archie, as the first internet search engine in 1990, was an index of FTP (File Transfer Protocol) files that allowed users to create simple requests for searching files [88]. Due to the increasing growth of the documents on the web, the indexing approach quickly was replaced by keyword search engines.

Keyword searching is the most common form of text search on the web by creating text queries and retrieving information using these keywords. The keyword search engines have been widely used on the web 2.0 to explore the documents by considering the links to/from a web page as a method of determining relevance [80]. Keyword search engines have been known as syntactic search engines because they depend on keywords as text in their queries [70]. Some examples of syntactic search engines include Google, Yahoo, Ask, and Msn [88].

Inline with keyword search engines, metasearch engines target to pass queries simultaneously to multiple search engines and then collect and integrate their results to a single result listing at the same time avoiding redundancy [89, 90]. For example, Metacrawler [91] combines the results from some of the search engines such as Google, Yahoo, Ask, Msn instead of crawling the web and maintaining its own index of documents. In contrast to metasearching which is based on just-in-time processing, federated searching is based on just-in-case processing [92]. Federated

searching combines a large amount of data into a single repository that can be searched [93]. For example, Google Scholar as a federated searching system provides a single user interface for searching across scholarly information [92].

Consider the case of finding and ordering food on the web through the keyword search engines. The keywords such as "order food online", "restaurants in Innsbruck", "free delivery" have been required to be searched by users looking for free deliverable meals in Innsbruck and then only the web pages including the keywords have been returned. Next, the users had to check the returned pages one by one to find their desired restaurants and finally order the foods.

2.4.1 Current State of the Art and Related Works

Keyword search engines paved the way for the next generations of search engines. It is difficult to get an accurate result in keyword search engines as they do not know the exact meaning of the keywords used. So, polysemy words and synonym words lead to false positive and false negative, respectively. With the massive amount of data available on the web, a simple keyword text search is an ineffective solution, and intelligence should be embedded into search engines. Recent efforts are being made to provide an effective way to search the web that will be treated in Section 2.5 as semantic search engines.

The history and rise of keyword search engines are studied by Seymour et al. [88]. Here, various index-based tools and keyword search engines, including Archie, Gopher, Veronica, Judgehead, Aliweb, Altavista, Ask Jeeves, and Northern Light, continue to Google, Yahoo! and Bing are described [94].

In Rahman's survey [95], the major challenges, issues, and downsides of keyword search engines have been discussed in detail. Then, the

paper has introduced different non-keyword based approaches, including semantic search, concept-based search, exploratory search, content-based search, open domain QA and a computation knowledge engine to address the keyword searching obstacles. The survey presented by Selvan et al. [96] offers a review of the ranking algorithms that the keyword search engines use. According to the paper, the ranking algorithms are categorized into three categories, including link analysis, personalized web search ranking, and page segmentation algorithms. Further, the paper gives an overview of the most prominent ranking algorithms of each category. The Page Rank (PR) [97], Hyperlink-Induced Topic Search (HITS) [98], and Focused Rank [99] as the main link analysis algorithms, the integrated page ranking algorithm as the key personalized web search ranking algorithm, fixed-length page segmentation, document object model-based page segmentation, vision-based page segmentation and combined approach segmentation as the major page segmentation algorithms have been discussed in the survey.

According to studies by Tokgoz et al. [100], Hussain et al. [101] and CheshmehSohrabi et al. [102], Yahoo outperforms Google in image retrieval (users use one or more keywords to retrieve the relevant image or images they are looking for) while studies by Uluc et al. [103], Cakir et al. [104] and Adrakatti et al. [105] reveal contradictory results on the effectiveness of image search engines and indicate the outperformance of Google in image retrieval as compared with Yahoo.

2.4.2 Process Model

Although different keyword search engines have various process models, they are essentially composed of three main components including information collection component, indexing component and ranking component as shown in Figure 2.3:

1. Information Collection Component: web crawlers are responsible for collecting information. A crawler (also called spider or bot) can be assumed as a program that continually browses the web to collect new pages. The collected pages are used in the indexing in the next component.

2. Indexing Component: the collected web pages are stored, organized, and indexed in this component to improve the speed of retrieval. When a page is in the index, it is in the running to be displayed as a result of relevant queries.

3. Ranking Component: this component provides the pieces of content that evaluate the desired pages for the query, which means that results are ordered by most relevant to least relevant. The ranking algorithms which can be applied in the ranking component are described in Section 2.4.4.

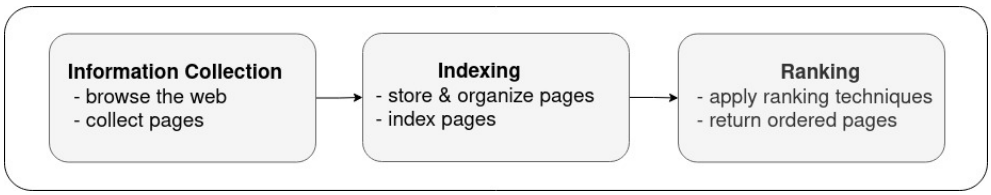


Figure 2.3: Keyword search engine process model.

2.4.3 Data Preparation and Representation

Keyword search engines have been broadly applied in the first and second generations of the web. The web 1.0 can be considered as the first generation of the web, which was ready-only, static, and somewhat monodirectional. The web 1.0 included websites with static HTML (HyperText Markup Language) pages that updated infrequently. The web 2.0, as the second generation of the web is a read-write web with blogs, RSS (Really Simple Syndication), wikis, mashups, tags, folksonomy, and tag clouds as its major technologies and services [80]. Therefore, the documents on the

web, either HTML or XML (Extensible Markup Language), can be considered the primary underlying resources for the keyword search engines, which are generally based on the occurrence of words in the documents and do not determine the intent and contextual meaning of the words.

2.4.4 Common Methodologies

With the increasing number of documents, search engines have the aim to find the best ranking order to satisfy user's query [106]. This is a task done by ranking algorithms. Ranking models assign scores to the documents representing relevance and similarity between the documents and the user queries. In general, the existing ranking techniques can be organized into three groups, as follows:

1. Content-based ranking: these approaches rank the relevant pages based on pages' content and keywords. Firstly, the words from the user query are stripped down to the root. The root words and their synonyms are considered for the construction of a dictionary. Then the keywords of each page are compared against the dictionary. Accordingly to the matches found, the relevancy of each particular page against the user query is computed as the ranking score.

2. Usage-based ranking: these approaches aim to rank web pages based on users' past navigation and retrieval patterns. A ranking score is assigned to each page which indicates how often they are viewed on the web. Thus, it determines the page's relevancy by its selection frequency. Ranking based on usage only can not guarantee precise results due to the other indications such as time spent on reading the page, the number of times the page was saved/printed or added to the bookmarks, and the actions of following the links of the page are neglected.

3. Link-based ranking: these approaches compute the ranking scores based on the links between the pages. For example, from the fact that a page has many links and references, it is derived that it must have something interesting to express. The link-analysis-based algorithms are generally computed offline, even before receiving any query from the user. Here, the popularity of pages is calculated by building a graph using a set of nodes and analyzing the existing links in it. The PR algorithm and the HITS algorithm are the most common examples of linked-based ranking algorithms.

2.4.5 Categories

In general, keyword search engines are divided into the following categories:

1. Web content mining: these search engines mine the content of web pages to extract the result by performing different mining techniques and shrink the search data, which become easy to find required user information [107].

2. Web usage mining: according to the log information stored during user interactions while surfing the web, user navigation patterns are discovered. Then the discovered patterns are applied to rank and fetch the desired pages against user queries [108].

3. Web structure mining: the main idea behind web structure mining search engines is to discover the structure of web pages based on the hyper links among them, create a structural summary and finally use the created the structural summary to extract the pages of given keywords [107, 108].

2.4.6 Summary

Keyword searching is the most common form of text search on the web. The keyword search engines, namely syntactic search engines, create the text queries and retrieve information using the keywords. There are three key steps to how most search engines work, including crawling, indexing, and ranking. To return the web pages in an ordered manner, web page ranking methods are applied, which can rank the pages in order of their relevance based on ranking algorithms. The different ranking techniques used in the keyword searching engines are divided into three approaches: content-based ranking, usage-based ranking, and link-based ranking. The PR and HITS as the link-based ranking algorithms are widely used in search engines. Web content mining, usage mining, and structure mining are assumed to be the major groups of keyword search engines based on their applied ranking techniques. Since the keyword search engines can not intelligently understand the context of what is being searched, they cannot be seen as an intelligent and effective solution to retrieve the information on the web.

2.5 Semantic Search Engine

Currently, most of the web's content is published so that humans can read it, and machines can identify where the main parts of the content are a text, a picture, or a link. However, machines cannot understand the semantic meaning behind them. Moreover, the increasing volume of data and web pages on the web has shown that traditional search engines are less suitable to provide the correct answer for users' questions. For example, a search engine retrieves information from the web, based on syntax matching of keywords provided by a user. This means that a keyword-based search engine will retrieve hundreds of results, which are

mostly not relevant to the query.

Furthermore, some limitations of traditional search engines are the lack of a structure when representing information (information is presented in a non-machine-readable format), heavy usage of computational power, and insufficient quality when answering long and complex queries. To overcome these problems, semantic search is required to improve search accuracy by consuming machine-readable web pages and semantic annotations.

With the growth of semantic web, semantic search engines have been fulfilled to search on the web [109]. The primary idea of semantic search stems from a data searching technique in which a search query intends to fetch keywords and specify the intent and contextual meaning of the user's query. Semantic search engines return relevant data to answer complex queries, which are restricted to particular entity types. For example, a query "Innsbruck restaurants serving Italian pizza" must be split into different entities such as "Innsbruck", "Restaurant", "Pizza" that represent a city, place and food, respectively. In other words, semantic search enables searching for entities (e.g., things, persons, places) instead of just strings as keywords. So, in the case of table order, semantic search improves traditional search by relying on data that come from the semantic web and are supported by semantic web technologies [109, 110].

2.5.1 Current State of the Art and Related Works

In this section, we discuss work that has been done on semantic search engines. Ilyas et al. [111] propose an abstract conceptual process model for semantic search engines, which should consist of ontology manager, web crawler, query builder and preprocessor, and inference engine. In the work presented by Sánchez-Cervantes et al. [112], a meta-search

system, called LINDASearch, is introduced to provide information about the well-known open linked data projects (e.g., DBpedia). LINDASearch aims to overcome several limitations, including faceted navigation and data unification over various domains, time optimization, and results' scalability. Sahu et al. [113] compare several search engines, for instance, evaluating search queries and their time on retrieving the answers, as well as the precision of the answers. The authors state that Google has the best performance, followed by Yahoo and Bing, respectively. Hussan [114] and Jain et al. [115] survey semantic-based search engines and point out their pros and cons. The authors state that all the surveyed semantic search engines use mainly semantic web technologies. They are more accurate than keyword-based search engines. However, semantic search engines are still complex to be implemented. For instance, semantic search engines must focus on hybrid approaches that combine various ways of interactions, leveraging their advantages [82].

Furthermore, Uren et al. [82] focus on reviewing the user experience of semantic searches. They describe four ways a user can interact with semantic search systems: keyword-based, form-based, view-based, and natural language-based. Some limitations are mentioned for form-based and view-based approaches, such as those that do not allow relation-based search and have low performance on showing large ontologies on an interface. In addition, the keyword-based search is limited to mainly using syntactic matching techniques, and natural language-based search provides means to formulate long-tail queries yet require more user interactivity.

Additionally, some limitations for semantic search engines are: the knowledge acquisition process limits the powerfulness of semantic search engines [116], and furthermore, there is some work to be done on matching and ranking methods used on semantic search engines, which mostly

reuse information retrieval techniques [82]. Last but not least, search engines must rely on high-quality knowledge sources [117]. However, it is not clear to which degree knowledge sources are correct or complete, e.g., Google's search engine relies on Wikidata, which might contain errors, duplicates, or missing values.

2.5.2 Process Model

There are several semantic search engines developed from the need for efficient, accurate, and scalable search on the web, e.g., some instances of semantic search engines include Kosmix, Kngine, Hakia, Cognition, Falcons, Lexxe, Scarlet, Sindice, Swoogle, SWSE (Semantic Web Search Engine), and Watson [115, 118, 119], while all these semantic search engines are developed following specific use cases and are based on different assumptions, a mutual understanding of the model that relates them to each other exists. For instance, (a) semantic web documents discovery, (b) indexing, (c) analysis, and (d) interface are the components of Swoogle [120], (a) Resource Description Framework (RDF) crawler, (b) document analysis, (c) vocabulary identification, reasoning, and indexing, (d) summarization, and (d) interface are modules of Falcon [121], and (a) Crawler, (b) Query detection and extraction, (c) ontology analysis, (d) query indexing, (e) query processor, (f) ranking, and (g) interface are components of Hakia [115].

The semantic web allows structuring content on the web to enable machines to perform tasks that need a level of data interpretation. Semantic search engines are mainly based on semantic web technologies, including standard data modeling, syntax, schema, and query languages and protocols. These technologies make semantic search engines work. In the following, semantic search engines process model (see Figure 2.4) is described:

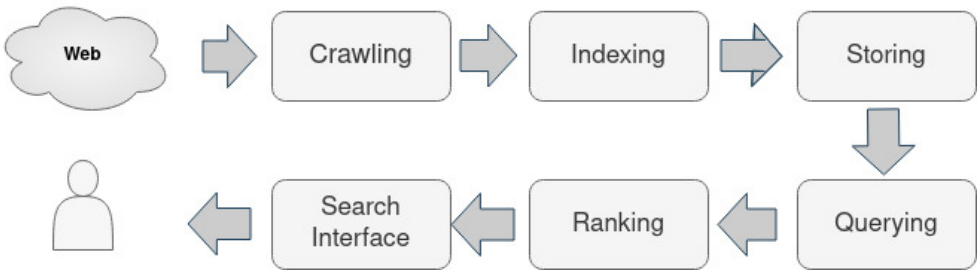


Figure 2.4: Semantic search engines process model.

1. Crawling: a crawler aims to discover documents and collect them. Like classic search engines, semantic search engines collect data using crawlers, for instance, crawling RDF documents embedded on websites. For that, more specialized crawlers have been developed [122], which provide various methods for crawling data; (1) direct URL (Uniform Resource Locator) crawling that bootstraps the crawling process, (2) Google-based crawling, which retrieves hyperlinks directly from Google search engine, (3) extracting and following hyperlinks limited to a certain depth and threshold, (4) RDF crawling that fetches semantic annotations embedded on webpages [111], furthermore, (5) crawlers that combine previous approaches, for example, using Google-based crawling to retrieve hyperlinks for most common words used in a specific language (e.g., English language).

2. Indexing: the indexing process analyses the retrieved documents from the crawling process. For instance, the vocabulary used and the relationship between resources and metadata about the retrieved documents (e.g., last modified) are analyzed. For instance, Swoogle uses a rational surfer model to rank retrieved documents [115].

3. Storing: the indexed documents are later on stored in a knowledge base in the form of graphs, which contain triples (subject, predicate,

object). The goal of storing the data using semantic technologies is to facilitate the answer of complex queries [115]. For instance, there are several graph database systems, such as GraphDB[2] and Neo4J[3], which support semantic queries.

4. Querying: after the indexing and storing processes have been done, the knowledge can be queried, while querying to traditional search engines return documents, the result of a semantic web search query is richer than just simple documents. Semantic search engines return a representation of an entity (note that an entity or a resource includes web pages, parts of a web page, devices, people and more) (i.e., classes, properties, and literal values). For this reason, the query system provided by a semantic search engine should be able to perform complex search queries, e.g., a user can express the context for a term that he or she is looking for, and the semantic search engine can disambiguate the term [111].

5. Ranking: the ranking process runs semantic analysis and concept match between the user's query and the output produced by the semantic search engine [115]. The ranking process evaluates several semantic and statistical metrics, e.g., context and popularity, which improve semantic search engines' algorithm [123]. For instance, Anyanwu et al. [123] propose SemRank that applies several techniques (e.g., semantic association) for analyzing and ranking relationships between two instances in a knowledge base.

6. Search Interface: "the proof of the pudding is in the eating", providing means to access the knowledge base is a crucial part of a semantic search engine. For instance, it is essential to provide means that facilitate access to knowledge for humans and machines.

[2] `https://www.ontotext.com/products/graphdb`
[3] `https://neo4j.com/`

Services like REST (Representational State Transfer) service API (Application Programming Interface) [120], keyword-based, form-based, view-based, or natural language-based semantic search systems must be supported [82].

2.5.3 Data Preparation and Representation

A semantic search engine collects, indexes, and stores richly structured data sources to later provide querying mechanisms to explore and retrieve knowledge. Semantics enables machines (e.g., semantic search engines) to understand the represented data better and discern the entities and their relationships.

Currently, most of the web's content is in natural language text, which raises an open question of whether computers will ever become as fluent as humans in understanding natural language text to interpret it the way humans would, for example, understanding entities and relationships between them. In order to overcome this problem, semantic technology techniques for retrieving and transforming structured and unstructured data into a knowledge base paradigm are required. For instance, various standards, such as RDF[4], RDF Schema (RDFS)[5], and Web Ontology Language (OWL) [6], were developed for the syntax and data model [124].

RDF has been published as a standard model for data interchange and proposed as a graph-based data model. In RDF, a document states that particular entities have properties with specific values. These are known as triples (subject, predicate, object). For example, the "Innsbruck Restaurant's phone number is +41 672343247" statement can be expressed as follows: a subject denoting "Innsbruck Restaurant", phone number as the predicate or property, and the object as value "+41

[4] `https://www.w3.org/RDF/`
[5] `https://www.w3.org/2001/sw/wiki/RDFS`
[6] `https://www.w3.org/OWL/`

672343247".

Schema.org[7] is the most widespread vocabulary and the de facto standard for annotation of data on the web. It is supported by the major search engines Bing, Google, Yahoo, and Yandex since 2011. Furthermore, Schema.org vocabulary, along with the Microdata, RDF in Attributes (RDFa), or JavaScript Object Notation for Linked Data (JSON-LD) formats, are used to markup not only data but also content and services on the web.

Semantic annotations have gained attention since the introduction of Schema.org, which empowers web search on a global scale. Search engines can recognize semantic annotations because they mark up websites' content and are embedded on websites. Fensel et al. [125] state that semantic annotations are the basis for building a knowledge graph.

Knowledge graphs, also known as knowledge bases, are large semantic nets that integrate diverse sources to represent knowledge in target domains [125]. Google's knowledge graph, launched in 2012 initially to improve Google's search results, boosted the adoption of knowledge graphs. For instance, large technology companies, including Amazon, Facebook, Google, Microsoft, and many more, have knowledge graphs, and have invested in their curation with the purpose to improve their web-scale services (e.g., knowledge graphs' content can be easily explored and analyzed via semantic search engines) [126].

Technologies that empower semantic search engines are already in place. The next step is to promote their adoption by everybody who produces content on the web. The performance of semantic search engines will increase as more machine-readable data become available on the web.

[7] `https://schema.org/`

2.5.4 Common Methodologies

There are several semantic search engines developed from the need for efficient search engines, while all the semantic search systems differ, some common methodologies relate them to each other.

1. Knowledge acquisition: To effectively harvest structured and unstructured knowledge from the web, hybrid crawlers were used. For instance, crawling semantic annotations from websites, following links from the crawled semantic annotations, harvesting RDF/XML documents, and URLs using traditional search engines. Furthermore, several methods for transforming unstructured and semistructured data into structured knowledge are needed, e.g., converting comma-separated values (CSV) data into RDF.

2. Knowledge base construction: It summarizes methods, such as schema alignment, entity matching, and entity fusion, for integrating knowledge into a knowledge base. For instance, to detect duplicates (i.e., entity matching), it is necessary to compare every entity with each other, which is not recommendable for large knowledge bases [127, 128]. In this case, indexing techniques might help to reduce the number of comparison, e.g., some indexing approaches are: an ontology-based index [129] that stores the ontology graph, an entity-based index that takes into account the relationships between entities, and a textual-based index that considers triples (subject, predicate, object). Additionally, more index techniques are listed in the paper presented by Lashkari et al. [130].

3. Semantic search services: Approaches to capture and process search queries based on various techniques, such as entity ranking algorithms, e.g., ranking approaches can be classified into three categories [131]: entity, relationship, and document ranking. Further-

more, semantic search services comprises the coupling between the stored knowledge base and an ontology to support the generation of queries, e.g., traversing the knowledge base (or a part of it) and using query templates are common approaches used for refining/creation of queries.

4. Semantic search presentation: Common ways of interacting with semantic search engines involve a) services for machines like REST service APIs and b) user-friendly interfaces such as keyword-based, form-based, view-based, and natural language-based semantic search systems.

2.5.5 Categories

According to Uren et al. [82], semantic search engines can be categorized into four groups based on their user interaction mode:

1. Keyword-based semantic service: These search engines try to boost the performance of conventional keyword search engines by considering semantic entities that match query keywords. They translate query terms into semantic entities through typed links among entities on the data web to find more accurate and relevant information.

2. Form-based semantic service: Form-based semantic search services aim to guide users to make queries based on the information needed. They facilitate formulating semantic queries by translating ontologies' parts into forms, menus, and drop-down lists.

3. View-based semantic service: The services intend to help users construct queries and explore domains by ontology presentation and navigation. The considerable benefit of web-based semantic services is that users can easily understand the domain. The query

vocabulary and content classification scheme can be presented in intuitive formats.

4. Natural language-based semantic service: the main idea is to employ semantic markup and natural language processing techniques in question-answer systems. The service takes a query expressed in natural language and a given ontology as input and finds the answer from one or more knowledge bases that subscribe to the ontology. Therefore, they are more flexible than form-based and view-based semantic services and do not require users to learn the vocabulary or structure of the ontology to be queried.

Furthermore, Wei et al. [132] classify semantic search into six categories: (i) document-oriented search that is an extension of conventional information retrieval techniques and retrieves semantic annotations, (ii) entity and knowledge-oriented search, which improves the previous category by exploiting links between entities to retrieve additional knowledge, (iii) multimedia information search that allows retrieving semantically related multimedia (e.g., images), (iv) relation-centered search, which additionally pre-processes user's query to find out relations between query terms, furthermore, relation-centered search are most often exploit in QASs (see Section 2.6) semantic analytics that entirely takes advantage of knowledge bases to discover and interpret complex relations between entities, and (v) mining-based search, which infers new assertion based on a knowledge base.

2.5.6 Summary

Semantic search engines are the cherries on the cake! They show the benefits of using semantic web technologies on specific tasks. We discussed a general overview of the common components/processes of semantic search engines: crawling, indexing, storing, querying, ranking, and search

interface. It should be noted that current semantic search engines have differences in the way they implement the components. Additionally, we point out that the technology (e.g., RDF, Schema.org) that empowers semantic search engines is already in place, and we only need to promote their adoption. Furthermore, a classification of semantic search engines over user interaction mode, and task-oriented search has been discussed. Last but not least, we list some limitations or challenges that must be faced, such as heterogeneity of information (i.e., ontology quality), scalability (i.e., the rapid growth of content on the web), and quality (i.e., a trade-off between correctness and completeness of data).

2.6 Question Answering System

QASs can be viewed as an extension of search engines in the sense that they target finding precise answers to natural language questions, rather than returning a ranked list of relevant sources [133]. Since in natural language, the same meaning can be expressed in different ways and the same phrase having different meanings, QASs need to address lexical gap and ambiguity, respectively, [3, 134]. For example, the vocabulary used in a question can be different from the one used in the underlying knowledge source. The first QASs were introduced in the late 1960s and early 1970s to access data over databases [64, 133, 135]. With academic research, QASs have become a crucial topic and attracted massive attention over unstructured and structured data such as text documents and RDF knowledge graphs. Thanks to the development of the semantic web, a large amount of structured data has become available on the web, and the demand for QASs increases day by day [136].

Imagine the example of table order on the web through QASs. These search services allow users to express their questions in a natural language form such as “Which restaurants in Innsbruck serve Austrian cuisine on

Sundays for between €30 and €60?" and then the precise answers are returned to the users. Here, users do not get lost in massive resources on the web and only receive answers of their posed questions over knowledge resources.

QASs overlap in some aspects with semantic search engines and even can be considered as a kind of natural language semantic search engines which employ semantic mark-up and natural language processing techniques to provide effective and convenient query techniques for end users [82].

2.6.1 Current State of the Art and Related Works

With the extensive research of the past years on QASs, these systems are used in a wide range of application areas, including web communities, medicine, industry, culture, or tourism [64, 135]. The main focus of past surveys can be summarized as follows.

The survey presented by Dimitrakis et al. [135] obtains insights regarding the underlying knowledge source in QASs, where their knowledge source can be (1) structured data such as SQL (Structured Query Language) databases and knowledge graphs, (2) unstructured data in the form of text documents, or (3) mixtures of them. It also provides several aspects for structuring the landscape of QASs and labeled the main recent systems based on different aspects (knowledge source, types of questions, and domain type). Furthermore, the most common evaluation datasets have been grouped according to various criteria, including the domain's type, knowledge source, available tasks, and evaluation metrics.

Analyzing QA collections to provide better descriptions about the challenges of QASs has been the main focus in the work presented by Rodrigo et al. [137]. It has shown that QASs can hardly address various rewordings in questions and documents and infer information that is

not explicitly mentioned in texts. Further, a set of directions for future evaluations has been suggested.

The study conducted by Wu et al. [1] gives a comprehensive review of knowledge base QA approaches. The methods to answer the question over a knowledge base have been classified into semantic parsing and information retrieval. Semantic parsing is a high-level analysis that targets to convert unstructured natural language questions into logical forms or executable queries. Information retrieval aims to extract information from questions, detect candidates across the knowledge base, and finally find the most suitable answer among the candidates. The study has discussed the mainstream techniques of each category, similarities, and differences among them in detail.

2.6.2 Process Model

According to the type of the underlying knowledge sources, the process models of QASs can be classified into two broad categories, namely: document-oriented model and data-oriented model [66, 83].

The document-oriented model targets extracting the answer from plain text and employs traditional information retrieval techniques combined with machine comprehension methods [135]. QASs over documents adhere to a pipeline model with three main modules, including question analysis, passage retrieval, and answer extraction as shown in Figure 2.5 [66, 135], and can be described as follows:

1. Question analysis: as the first activity, questions are classified based on their type, leading to recognizing the expected answer types. Then named entities of input questions are identified, and their relations are detected. The last activity of this module aims at enhancing question phrasing through adding more descriptive information to increase the accuracy of the system, for example, using

WordNet as a lexicon in order to retrieve semantically equivalent information [66, 135].

2. Passage retrieval: to find relevant information in the underlying documents as source knowledge, this module employs information retrieval techniques for returning a ranked subset of the most relevant documents. Then the relevant documents are segmented into shorter units, namely, passages. Finally, the candidate passages are ranked according to features such as the number of question words in the passage and the number of named entities having the answer type in the passage. These features should determine the probability of containing the precise answer [66, 135].

3. Answer extraction: this module targets identifying the answer candidates from the ranked list of passages. Then the candidate answers are ranked according to features that reflect the probability of being the precise answer. To deal with the partial answers between passages or documents, it is responsible for generating the final answers and computing a confidence score that reflects the confidence of its accuracy.

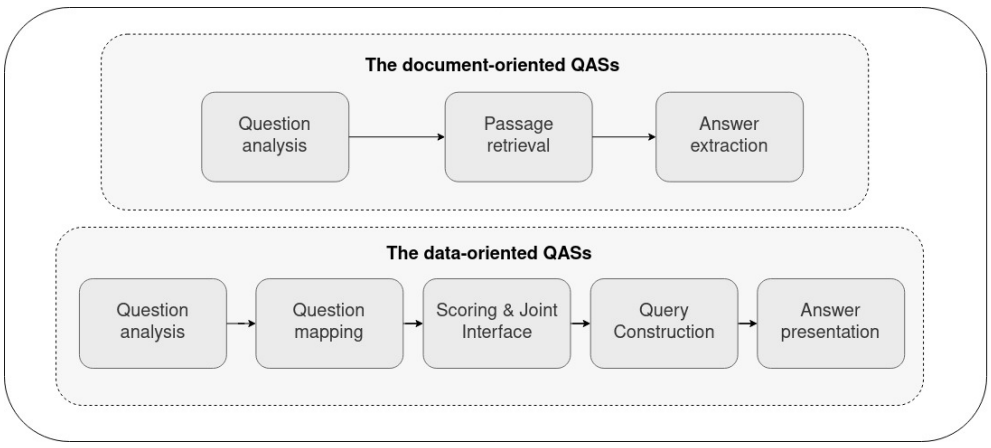


Figure 2.5: The document-oriented process model.

With the increasing maturity of structured data on the web, e.g.,

knowledge graphs, most QASs over structured data adopt a multi-component process model.

The major components of the data-oriented process model can be listed as question analysis, question mapping, scoring, and joint interface, query construction, and answer representation as shown in Figure 2.5 [133].

1. Question analysis: it aims at analyzing the input question linguistically and syntactically. The linguistic analysis of the question leverages part-of-speech taggers and parsers to capture the syntactic structure of the question, e.g., Named Entity Recognition. The semantic analysis targets identifying the question type and the focus question.

2. Question mapping: this component's main goal is to match question words or phrases to their counterparts in the underlying knowledge source, e.g., RDF knowledge graphs. Due to the lexical gap and the ambiguity between user questions and the underlying knowledge graph vocabulary, synonymy, hypernym and hyponym should be considered in computing similarities.

3. Scoring and joint interface: to select only one candidate among the candidate components, a scoring mechanism is required to score the candidates. The semantic similarity between the candidates and the question can be applied to achieve a scoring mechanism to define a matching score. Moreover, string matching combined with linear programming can be used.

4. Query construction: in order to transfer the question to an executable query such as SPARQL Protocol and RDF Query Language (SPARQL)[8], the approaches can be summarized into two groups:

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

(a) template-based approaches which map input questions to generated SPARQL query templates, (b) template-free approaches which aim at creating SPARQL queries according to the given syntactic structure of the input question.

5. Answer presentation: since the structured representation of answers is not intelligible for users, the answer presentation component follows some processing activities to transfer the RDF answers to a natural language form.

2.6.3 Data Preparation and Representation

The fundamental goal of QASs is supplying accurate answers to questions posed by users in a natural language form [135, 24]. The knowledge sources which QASs exploit in order to answer the user questions can be documents, data, and combinations of document and data [135].

Documents: the document-based knowledge sources have widely always received massive attention from by considering information retrieval techniques and recently deep neural network techniques [138]. The documents include plain texts, e.g., textual excerpts from Wikipedia, and require employing massive natural language processing and natural language understanding techniques to accomplish the goal of QASs [135].

Data: in the case of data, the underlying knowledge source exploits structured data. Since the amount of available structured data on the web consisting of RDF datasets keeps growing, RDF knowledge graphs have become a powerful asset for enhancing QASs. RDF knowledge graphs are huge collections of interconnected entities enriched with semantic annotations [139]. Due to the semantic relations of the data stored in knowledge graphs, applying complex natural language processing techniques is less than the document knowledge sources. Prominent examples of large-scale knowledge graphs include DBPedia [22], Yago

[140], Freebase [141], Wikidata [36] Google's Knowledge Graph, Facebook's Graph Search, Microsoft's Satori, and Yahoo's Knowledge Graph [106, 142].

Combinations: the combination of documents and data can be considered the hybrid knowledge representation. In this case, QASs target deriving answers using both a corpus and a knowledge graph [8]. Knowledge graphs are mostly incomplete, which leads to low recall values in QASs over KGs. On the other hand, the diversity of natural language makes document-based QASs difficult however a corpus may contain more answers than a knowledge graph [7]. Thus, the combination of knowledge graphs and documents (or corpora) can improve the accuracy of QASs, e.g., GRAFT-Net [7] and PullNet [8].

2.6.4 Common Methodologies

The research progress of QASs as a reasonably long-existing research field of computer science can be categorized in three groups [74]:

1. Traditional techniques. Frequently asked question and answer (FAQs) and rule-based methods can be assumed as the major traditional and straightforward approaches. In FAQs, a set of question and answer pairs are collected and stored as the dataset. Then answers are generated by searching the given question from the stored dataset. When the required query is found, the relevant answer is given back to the user [143]. The rule-based methods primarily used in QASs over knowledge graphs rely on predefined rules or templates to parse questions and provide logical forms [74]. The definition of rules or templates leads to limited scalability and the need for researchers to become familiar with linguistic knowledge.

2. Information retrieval-based techniques. Machine learning, more specifically deep learning, plays a key role in many aspects of infor-

mation retrieval systems [144]. A QAS over documents generally involves question and document representation steps, followed by a matching step to estimate the mutual relevance of the query and the document representations. A neural approach can affect one or more of these steps. In the case of QA over knowledge graphs, for a given natural language question, the named entities that reflect the main focus of the question are identified. Then the link between the extracted entities and the knowledge graph are specified. In the next step, a subset of the knowledge graph around the identified entities are extracted what its nodes are assumed as candidate answers. Based on the features extracted from the questions and candidate answers, the matching scores between the encoded answers and questions are calculated, and the final answer is selected. From the feature representation's perspective, the information retrieval-based techniques are divided into two groups including feature engineering and representation learning. In feature engineering, features are manually defined according to the questions' syntax information which basically fail to capture the semantic information of questions. In representation learning, questions and candidates are transformed to embeddings and then the embeddings are leveraged to compute the semantic matching and find the answer. The state-of-the-art QASs employ neural deep networks (e.g., memory networks, Convolutional Neural Network (CNN), Long Short-Term Memory (LSTM)) to generate better distributed embeddings for questions and candidate.

3. Semantic parsing-based techniques. Knowledge graphs based QASs can leverage semantic parsing-based techniques in order to answer user questions. These methods usually transform natural language questions into executable queries such as SPARQL queries or inter-

mediate query forms such as query graphs based on neural semantic parsing with high scalability and capability. Recent QASs take advantage of graphs to represent questions, namely query graphs. Basically, the query graphs can be generated based on predefined natural language processing rules or neural networks. Then, the query graph which basically have topology commonalities with knowledge graphs are mapped to the knowledge graphs to find the answers. Additionally, trees or high-level programming languages can be used to represent questions through sequence-to-sequence models and attention mechanisms [145].

2.6.5 Categories

QASs can be classified according to a wide variety of criteria. Among the various possible categorizations, this section presents a description of the categorizations based on the application domain and forms of answers generated [64, 136, 146].

According to the application domain, QASs can be grouped into two groups: open-domain and closed-domain QASs. Open-domain QASs focus on answering domain-independent questions within a huge knowledge source. In closed-domain QASs, the QASs answer questions under a restricted domain such as temporal, geo-spatial, medical, patent, and community [64, 136, 147].

The categorizations of QASs based on forms of answers include [66]:

1. Sentence/paragraph-answer based QAS: To answer some types of questions such as factoid or hypothetical ones, QASs return a single fact or a small piece of text as sentence/paragraph as the relevant answer.

2. Yes/no-answer based QAS: User questions are generally answered in the form of yes or no through verification and justification.

3. Multimedia-answer-based QAS: Answers are generated in different types of multimedia such as audio or video.

4. Opinionated-answer-based QAS: This system gives a star rating to an object as the answer.

5. Dialogue-answer-based QAS: These QASs are also known as dialogue systems that make efforts to answer users' questions in the form of a dialogue.

2.6.6 Summary

Many QASs have been emerging over documents and structured knowledge bases. According to the underlying knowledge sources, the process model of QASs can be classified into document-oriented model and data-oriented model. The main modules of the process model of QASs over documents include question analysis, passage retrieval, and answer extraction. Similarly, question analysis, question mapping, scoring and joint interface, query construction, and answer representation are assumed the basic components of the data-oriented process model. The text corpus and structured RDF data have been taken into consideration regarding data preparation and presentation in different QASs. The major techniques to answer natural languages questions consist of traditional techniques, information retrieval-based techniques, and semantic parsing-based techniques. Categorizing QASs based on domain involves open-domain and close-domain. The answer type criteria comprise sentence/paragraph-answer, yes/no-answer, multimedia-answer, opinionated-answer, and dialogue-answer-based QAS.

Although QASs have a long history and many approaches have been introduced, the accuracy of QA over unstructured and structured data still needs to be improved, and a significant amount of work is required

to make them more beneficial and practical in the real world.

2.7 Dialogue System

Basing on an essential skill of human beings, interaction via natural language is the most straightforward way humans can access information on the web. If humans can communicate through natural language, they can conveniently access the desired information [148]. Dialogue systems, as a subset of QASs intend to communicate with human users relatively naturally through a dialogue. Dialogue can be considered a conversation between humans and machines. The form of communication can be text, speech, images, video clips, and or sign language [77, 85, 149]. Since these systems have to be interactive, incremental dialogue processing is required to generate continuous dialogue. In incremental processing, processing basically starts before the input is complete and that the first output is generated as soon as possible [150, 151]. Apple Siri[9], Google Assistant[10] or Amazon Alexa[11] are some example of dialogue systems (more precisely chatbots) to support information search interactions. The dialogue systems can be applied to a wide variety of fields, including information-searching services via questions, virtual assistants to help users in daily tasks, e.g., scheduling appointments, and E-learning dialogue systems, e.g., train military personnel in questioning a witness [152]. For example, a goal oriented dialogue system for booking a restaurant would take several steps. First of all it would ask you what kind of food you desire, where you are located, what price range is acceptable, and the date and time you like to book a table. Afterwards it suggests you some restaurants. Finally it asks you if you are satisfied or want to make some changes.

[9] `https://www.apple.com/es/siri/`
[10] `https://assistant.google.com/`
[11] `https://www.amazon.com`

2.7.1 Current State of the Art and Related Works

Currently, the fields of dialogue systems range from small turn-based chat applications to large systems which can determine the emotions of the user and respond corresponding like Microsoft's XiaoIce [153]. There are also specific dialogue systems like the "Multi-modal dialogue system with sign language capabilities", which has been proposed for deaf people [149].

The survey presented by Chen et al. [75] explains recent advances from deep learning to build dialogue systems and discusses possible research directions for task-oriented and non-task-oriented dialogue systems in detail. According to the survey, end-to-end-based and pipeline-based approaches are the main directions for task-oriented dialogue systems, while neural generative models, retrieval-based models or a mixture of neural generative and retrieval-based solutions are considered the major approaches to developing non-task-oriented dialogue systems.

Mallios et al. [154] present a high level process model for dialogue systems with six main components: automatic speech recognition, natural language understanding, dialogue management, knowledge base, response generator, and text-to-speech synthesis. Additionally, it proposes a classification scheme of dialogue systems, including answering systems, semi-dialogue systems, and full-dialogue systems.

Chaves et al. [155] show an increase in users' expectations for formal and specialized discourse in using conversational agents. The findings contradict the results presented by Liebrecht et al. [156], who argue that users may not assign different roles to conversational agents in a human-agent customer service setting.

2.7.2 Process Model

Different dialogue systems may have different process models but in general the various models can be summed up in three layers as shown in Figure 2.6 [85, 153].

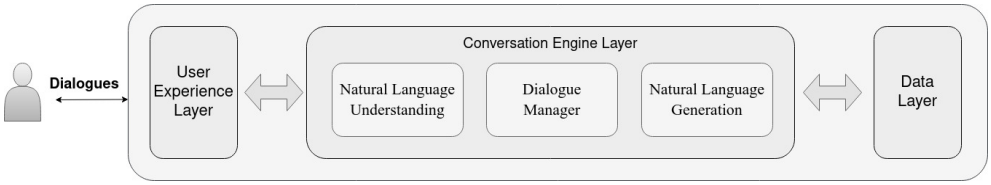


Figure 2.6: The process model of dialogue systems.

1. User experience layer: this layer is the user's input and the dialogue system's output layers. As mentioned previously, the input and output can be in the form of voice, text, images, video clips, and or sign language. The user experience layer can be connected to many platforms such as Facebook Messenger, WhatsApp, WeChat, Sina Weibo, Tencent QQ, Telegram, and Skype.

2. Conversation engine layer: the second layer is connected to the other layers, including the user experience and data layers. It analyses the user input and preserves the data in a useful manner in the data layer. Depending on the task of the dialogue system, the data is used to give the desired answer. For speech-based dialogue systems, a Text To Speech module is added at the end. The conversation engine layer consists of three main modules as shown in Figure 2.6 [85].

   (a) Natural Language Understanding Module: it takes a user query in a natural language manner and translates it into a semantic representation. If the input is not a text, it is converted to a string of words, then stored in the Data Layer.

   (b) Dialogue Manager Module: the semantic representation of the

user's text is taken, and a schematic representation of response is generated. Indeed, the module keeps track of the dialogue state and a dialogue policy learner, which decides what to answer to the user.

(c) Natural Language Generation Module: the semantic representation of the answer is translated to a natural language answer. Thus, it makes decisions regarding the information to be included, how information should be structured, choice of words, and syntactic structure for the message.

3. Data layer: all data, including collected conversational data, non-conversational data, and knowledge resources, is stored in the data layer as a set of data sources.

2.7.3 Data Preparation and Representation

Dialogue systems intend to interact with human beings and rely on a huge amount of different types of data, including unstructured, structured, and semi-structured data [75]. The key ability of these systems is to understand unstructured data as input and interpret it correctly and then provide the output across the underlying knowledge base, which are described as the following:

1. Unstructured data: this kind of data is not organized in a predefined manner. Text documents are the most well-understood instance of unstructured data which do not have any predefined data model.

2. Structured data: due to addressable elements in structured data, it is easy to be analyzed. It can be effectively organized in a formatted repository such as databases. The relational databases are

the most well-known model for holding tabular data as an example of structured data.

3. Semi-structured data: it lies between unstructured and structured data. However, the semi-structured data does not reside in a relational database, but it has some organizational features that make it easier to analyze. The XML data resources can be considered semi-structured data.

2.7.4 Common Methodologies

According to Chen et al. [75], different methodologies to develop dialogue systems can be summarized into the following varieties:

1. Pipeline methods: the main idea behind the pipeline methods is to define a pipeline structure including natural language understanding, dialogue state tracker, dialogue parser learning, and natural language generation. Natural language understanding is responsible for detecting the intent of users through classifying the intent into predefined intents or employing some techniques such as deep learning. The dialogue state tracker handles the input of each turn along with the dialogue history and returns the current dialogue state. Then the dialogue parser learning learns the following action based on the output of the dialogue state tracker. Finally, the response is returned based on the action [75].

2. End-to-end methods: the task-oriented dialogue systems can apply an encoder-decoder model to train the whole system. The model only adopts a single module and interacts with structured external databases. In end-to-end methods, dialogue system learning can be viewed as the problem of learning a mapping from dialogue histories to system responses [75].

3. Retrieval-based methods: the major focus of retrieval-based methods is message- response matching by employing matching algorithms to bridge semantic gaps between messages and responses. So based on the matching, a response from candidate responses is selected [157]. The early retrieval-based dialogue systems apply single-turn response matching. In these systems, only the message is used to select a proper response in each single-turn conversation, while multi-turn response matching has been mainly used in recent years, current messages and previous utterances are taken to select the response. Thus, the selected response is natural and relevant to the whole context [75, 158].

4. Neural generative methods: to develop non-task-oriented dialogue systems such as chatbots, neural generative methods can be basically adopted. The sequence-to-sequence models as the foundation of generative methods lead to keeping conversations active and engaging [75, 159].

5. Hybrid methods: recently, the neural generative and retrieval-based models have been combined to increase performance. Since the retrieval-based systems generally return accurate but straight responses, while generation-based systems often return fluent but meaningless responses, the combined approaches can have significant effects on performance [75, 160, 161].

Neural models such as CNN, Recurrent Neural Network (RNN), Hierarchical Recurrent Encoder-Decoder (HRED), memory networks, attention networks, transformer, deep reinforcement learning models, and knowledge graph augmented neural networks are mainly used in state-of-the-art dialogue systems [162], but this paper focuses on providing a generic picture from the methodologies and methods' prospective.

Recently, dialogue systems apply CNNs as a hierarchical feature extractor after encoding the dialogues [163, 164]. RNN models and their variants (e.g., gated recurrent unit (GRU), LSTM) as powerful learning models are utilized in dialogue-related tasks since the dialogues are not independent of each other and are not of fixed length [165]. HRED as context-aware sequence-to-sequence models are able to capture hierarchical dialogue features in dialogue systems [166]. To keep dialogues, memory plays an key role in dialogue systems. That is the reason why memory networks are fulfilled in these systems especially task-oriented systems [162]. Attention networks and transformers are employed in the state-of-the-art dialogue systems due to attention is able to catch the importance of different parts in the dialogues and transformer not only is a sequence-to-sequence model but also is a model to represent the dialogues [167]. Additionally, due to the agent-environment nature of dialogue systems, deep reinforcement learning models are applied to make improvements in these systems [168]. Knowledge graph augmented neural networks are basically used in the dialogue systems that rely on structured or semi-structured data such as knowledge graphs [169].

2.7.5 Categories

Conversational or dialogue systems are categorized into task-oriented and non-task-oriented groups [77]:

1. Task-oriented dialogue system: Furthermore, known as goal-oriented dialogue system, it targets to assist users in completing tasks in one or multiple domains by the end of the dialog. Some goal-oriented dialogue systems include restaurant reservation, flight ticket booking, and course selection advising, finding products. The major methods to develop task-oriented dialogue systems include the pipeline or end-to-end methods.

2. Non-task-oriented dialogue systems: It also called chatbot can be defined as software to provide extended conversations and mimic the unstructured conversations or 'chats' characteristic of human-human interaction [77]. Chatbots aim to maximize user engagement and interact with a human to provide reasonable responses and entertainment. Typically they focus on conversing on open domains. However, the none-task-oriented systems have been basically designed for entertainment. They also intend to gain practical targets over time like task-oriented chatbots [75, 77]. The widely applied method to none-task-oriented dialogue systems are the retrieval-based, neural generative-based, and hybrid methods.

2.7.6 Summary

Dialogue Systems as a subset of QASs allow the user to converse with a machine using natural language. The major layers of a dialogue system include the user experience layer, conversation engine layer, and data layer. The conversation engine layer consists of the natural language understanding module, the dialogue manager module, and the natural language generation module. The data layer relies on a massive amount of various sorts of data, including unstructured, structured, and semi-structured data. The research progress of dialogue systems can be summarized into pipeline methods, end-to-end methods, retrieval-based methods, neural generative methods, and hybrid methods. The task-oriented and non-task-oriented dialogue systems such as chatbots are the most significant types of dialogue systems.

2.8 Chatbot

Chatbots are the most straightforward kind of dialogue systems designed for imitating human-human interaction [77]. Areas that significantly benefit from chatbots are entertainment, health care, marketing, supporting system, and customer service. Especially in the customer service domain, chatbots reduce costs by handling multiple users simultaneously with a 24/7 availability.

Imagine a chatbot embedded on a social media platform helping users book a table in your restaurant. Based on the information provided by your profile, the chatbot needs only a minimal amount of additional information to fulfill the table reservation. For example, the user's name can be taken from the profile, and only the desired date and time need to be provided. If successful, the chatbot immediately returns a confirmation. Besides, the chatbot may send a notification when the user has to leave to reach the restaurant in time (based on the current location, if the user uses an app and allowed location features).

The era of chatbots started a while ago with ELIZA [170] in 1966, then PARRY by Kenneth Mark Colby in 1972 [171], the first chatbot with personality, and the well-known intelligent digital assistants Amazon Alexa, Apple Siri, Google Assistant, or Microsoft Cortana, we use nowadays [172]. The main idea behind chatbots is the automation of services via conversational interfaces. Chatbots may be not only a tool but also a friend. However, they still struggle to understand the meaning and undertones of conversations.

Nowadays, most of the existing chatbots are acting on a closed domain [173]. Closed domain implies that they are designed for a particular task, e.g., booking a hotel room or reserving a table at a restaurant. Examples of open domain chatbots are Amazon Alexa, or Apple Siri, able to answer all kinds of questions [173].

Since we are in the middle of a pandemic, chatbots are also usable for sharing up-to-date information quickly and concisely. In such a scenario, short and precise answers from a trustworthy resource are essential to answer all kinds of questions [174].

2.8.1 Current State of the Art and Related Works

Chatbots have a pretty long history starting with the first chatbot ELIZA in 1966. Since then, many new approaches for designing and implementing chatbots have evolved. An overview of the history of chatbots is given in the work presented by Weizenbaum [170].

In addition to the history of chatbots, the surveys presented by Jurafsky et al. [77], Nimavat et al. [173], Adamopoulo et al. [175] and Deshpande et al. [176] classify chatbots based on different characteristics, be it the knowledge domain, their goal, or the communication medium. Besides, essential concepts and methodologies for processing natural language input and natural language generation are introduced. Every chatbot follows a general process model of user input, input processing, dialogue management, chat engine, and response generation. Suta et al. [177] cover the same topics focusing more on machine learning approaches used by chatbots.

From the technology anxiety perspective, the results of studies by Kim et al. [178], Lee et al. [179] and Yang et al. [180] show that technology anxiety is a reducing factor for relationships between the chatbot quality dimensions (i.e., understandability, reliability, assurance, and interactivity) and users' technology adoption. However, studies by Chin et al. [181] and Li et al. [182] argue that the higher levels of technology anxiety, the stronger the relationships between the chatbot quality dimensions and users' technology adoption.

For the design and implementation of chatbots, the user experience is

an essential part. In the study conducted by Jain et al. [183], first-time chatbot users are interviewed about their experience using different kinds of chatbots. As an outcome, users prefer human-like natural language chatbots or chatbots with an engaging experience. Those findings are usable for the design of new chatbots.

2.8.2 Process Model

In general, the process model of chatbots consists of four major components: Natural Language Understanding (NLU), Dialog Management (DM), Chat engine, and Response Generation [77, 173, 175, 176]. Figure 2.7 presents an overview of the process model and how the components are connected.

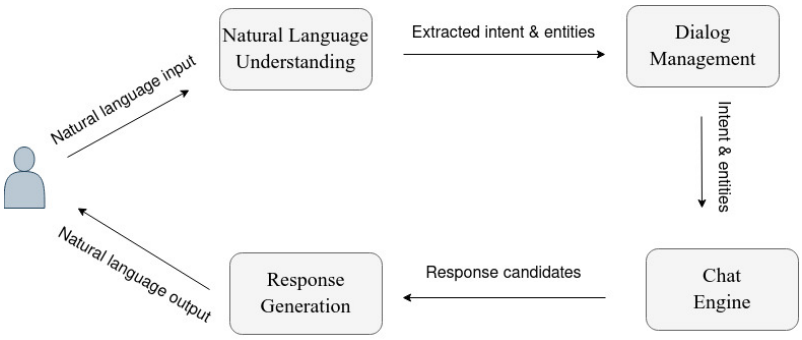


Figure 2.7: Chatbot process model.

Everything starts with a user input that needs to be processed by the system. The user input reaches the NLU component to be analyzed. In this component, the input is parsed, and the user intention and associated information are extracted. Besides, an NLU might analyze a user's sentiment, positive, negative, or neutral [177]. The output of this component is the identified intention (what the user wants to achieve) and associated information. For example, the intention of the question "What is the weather going to be in Innsbruck tomorrow?" is to get weather information. Associated information is the location ("Innsbruck") and the date ("tomorrow"). That information is passed to the Dialogue Man-

agement component to be processed further. The Dialogue Management Component keeps track of the current conversations, and their contexts [173, 175]. This implies the current intent and identified entities. Besides, this component ensures that all required information is available, follow-up questions are sent to the user if any information is missing. When all information is given, they are forwarded to the chatbot engine.

Each chatbot's core is the chatbot engine, responsible for executing the user's intended action. The underlying data, be it a database or an external API, is accessed to retrieve the information sent to the user. Based on the found information, single or multiple possible responses are sent to the Response Generation.

The Response Generation component uses Natural Language Generation (NLG) approaches to construct a personalized response using natural language based on the responses. Personalized means a proper writing style and emotions [177]. Additionally, the response should be grammatically correct, and the user should not recognize that the opponent is a chatbot rather than a human [184]. Therefore, the response should also include some emotions and imitate human-like behavior [185].

2.8.3 Data Preparation and Representation

The main goal of chatbots is to either help a user fulfill a particular task (closed domain) or imitate human-human conversations (open domain). Both ways need data that can be given to the user. Data used to power chatbots can be either unstructured, semi-structured, or structured [177]. Some chatbots store and host the data on their side, whereas others retrieve the relevant information in real-time from APIs. Retrieving APIs in real-time requires a connection from the chatbot engine towards the service to access. The connection to those web services can be hardcoded for simpler chatbots that require access to a small number of APIs only.

For chatbots handling large amounts of web services or when web services change very often, a description language on how to map the incoming parameters from the NLU component to the request's parameters to the web service is required. Web API Annotations with Schema.org Actions (WASA[12]) is such a description language. WASA is based on schema.org actions allowing to describe web services as a set of actions to be taken (e.g., search).

Storing chatbot data requires a database, such as a document store, a relational database, or knowledge graphs repository. A knowledge graph is an optimal solution for more complex chatbots accessing a vast amount of knowledge [125, 186]. Due to the flexibility of knowledge graphs, integrating heterogeneous sources is not an issue [125]. A vocabulary for defining the data stored in a knowledge graph is often schema.org [125]. Schema.org is the de facto standard for annotations on the web founded by the large search engine providers, such as Google, Bing, Yahoo!, Yandex.

2.8.4 Common Methodologies

Standard methodologies for chatbots mainly concern intent detection and extraction of entities. Besides, answer generation is essential for chatbots. In the following, methodologies and algorithms based on the classification presented by Hussein et al. [76] are considered:

1. Parsing—This category extracts meaningful information from the textual input of the user. The grammatical structure of a sentence is used to extract keywords that are then matched against the stored data to find the appropriate response. Semantic parsing is a more advanced technique for converting the input sentence into a representation that machines can process. For example, Di-

[12] `http://wasa.cc/`

alogflow recognizes relevant information based on a predefined set of training phrases containing placeholders for parts of the sentences containing relevant information. Those placeholders identify a specific type of information (e.g., location information or name information) that can be used to generate queries to the underlying database by using templates. This approach allows an explicit definition of what happens with the relevant information. However, it requires a lot of manual work to set up the chatbot properly [76].

2. Pattern Matching—This approach is most commonly used. It defines a set of handcrafted pattern-template pairs. Whenever a pattern matches the input, the template is used to return a response to the user. This approach is mainly used in QA chatbots and is very flexible for creating conversations. However, all possible patterns are built manually, which is not scaling. Due to the scaling issues, the responses are predictable and are not manifold, more repetitive [76].

3. Ontologies—Ontologies are used to represent domain knowledge and make it explorable by the chatbot itself. An advantage of this approach is that the chatbot can use reasoning to detect relationships between concept nodes used in the conversation [76].

4. Markov Chain Model—The Markov chain model is a probabilistic model modeling probabilities of state transitions over time. There is a fixed probability for the following states based on a given state. A chatbot using this model produces outputs based on those state transitions. The chatbot constructs probabilistically more suitable responses. This model is mainly used for chatbots that entertain users by imitating simple human conversations. Markov chain models however do not work well on complex conversations [76].

5. Neural Networks Models—These models allow more intelligent chatbots. The research trend for using artificial neural networks for chatbots is a generative-based approach where chatbots dynamically generate the response to the given user input. Neural network models are learning algorithms used in machine learning and can be supervised and unsupervised. Subclasses of artificial neural networks models used for chatbots are RNNs, sequence to sequence neural model, or LSTMs. A major problem with these artificial neural networks is that they are not accurate yet. Therefore, prominent virtual assistants like Alexa, Siri, and Cortana rely on a semi-rule-based approach [173].

2.8.5 Categories

Chatbots can be classified based on different categories [77, 173, 175, 176]. However, a chatbot does not entirely fall into a specific category but is part of multiple categories depending on different characteristics.

Starting with the build method, open-source platforms and closed platforms are distinguished [175]. Open-source platforms allow a broader community to develop the platform further and check how the platform is working. A closed platform acts as a black box, and further developments depend on the company maintaining the platform. As a communication medium [177], chatbots can either be integrated on messaging platforms (e.g., Facebook Messenger, Slack, WhatsApp) or used on smart home devices (e.g., Amazon Alexa, Apple Siri, Google Assistant) as so-called skills.

We follow with moving away from technical aspects to the concept behind the chatbot themselves and its goals. For goals, [175], chatbots can be classified as informative, chat-based, or task-based. Informative implies that users can ask for information about specific topics or get

general information (e.g., health information during the corona pandemic [174]). Chatbots with a chat-based or conversational goal act like another human being imitating real conversation with users. These can be seen as a friend of the user. Task-based chatbots are designed to perform particular tasks, e.g., a chatbot on a restaurant website that mainly handles table reservations. Usually, such chatbots also provide some information about the restaurant, but their main task is to allow users to reserve tables.

Some chatbots do not act autonomously but include a human in certain parts. This category is referred to as "Human-aid" [175]. A chatbot falls into this category if at least one part of the chat flow involves a human.

The core functionality of a chatbot is the processing of inputs and the generation of responses [175]. In this category, several types are distinguishable. The first chatbots (ELIZA and PARRY) were based on pattern matching approaches. A given input was matched according to the inputs stored in the internal database, and the predefined response for the matching input was returned. Disadvantages of this approach are that there will always be the same answer for the questions, no human touch, and previous conversations are not stored. Rule-based approaches are pretty similar to pattern matching. Based on a fixed set of rules, responses are generated. User input is mapped to the associated responses. Retrieval-based chatbots do not store the data themselves but query and analyze external sources using APIs. For example, a weather chatbot consults the API of predefined weather services. Generative or corpus-based systems are trained on a vast dataset of human-human conversations. Answers are generated based on answers from previous conversations. This approach is more human-like, uses machine learning algorithms and deep learning techniques but is challenging to build and

maintain. Especially for the setup of such bots, a vast amount of training data is needed. Besides the previously mentioned types, there are hybrid approaches that combine rules and machine learning approaches. In such cases, the management of the conversation flow is defined by rules, and the responses are generated using natural language processing approaches.

Regarding the knowledge domain [173, 175], two categories are used to classify chatbots. Closed domain chatbots focus on particular topics or specific tasks. Those chatbots maybe have some small talk content but mainly focus on the topic for which they are designed. Open-domain chatbots try to cover as many topics as possible.

Finally, chatbots can be distinguished based on the service they provide [173]. Suppose a chatbot is designed for a specific task, be it reserving a table in a restaurant or more as a communication medium to automate services. In that case, they are classified as interpersonal. Those chatbots do not act as friends of a user but more as a tool to achieve a specific goal. Intrapersonal chatbots are designed to be a friend of a user. They should understand the user like humans do. In the third category, there are inter-agent chatbots. Here, a protocol for the communication between chatbots is needed. An example of this type is the Alexa-Cortana integration.

2.8.6 Summary

Chatbots are a subclass of dialogue systems and the most superficial subtype of those. The process model can be split into four major components: the NLU component that identifies the user intention and extracts the entities, the Dialogue Management component taking care of the conversation flow and follow-up questions when information is missing, the chatbot engine itself accessing the underlying database or external API to

retrieve the responses that are relevant for the user, the NLG component creating a response for the user in a way a human would communicate. The presented methodologies mainly target NLU and NLG since those are the most crucial parts of a chatbot.

However, there are still many issues that need to be tackled by further research to make chatbots more human-like. This includes context awareness, diversity of responses, and chatbots should get a stronger personality.

2.9 Future of Search on the Web

As the primary tool for users to access information, the importance of search on the web can not be overestimated. To sum up, the main limitations or challenges of each type of the described search services are summarized as following:

1. Keyword search engines fail to (1) disambiguate words and return irrelevant results (false positives), and (2) turn up related materials that do not specifically use the search keywords (false negatives).

2. The stability of the semantic web languages and continuous development of ontologies are key challenges in semantic web search engines. Furthermore, the performance of semantic web search engines relies on knowledge sources, which might contain errors and missing values.

3. Despite a lot of progress in QASs, lexical gap and ambiguity are still main challenges in these systems either documents-based, data-based, or mixture of document and data based knowledge sources.

4. In addition to lexical gap and ambiguity, dialogue systems need to

tackle with incremental processing due to these systems are naturally interactive.

5. Additionally to the above challenges, remembering the context of a conversation (or even understanding in some cases) needs to be improved in chatbots.

Although keyword search is still important, semantic search is on the rise since a search query aims to not only find query terms but to determine the intent and contextual meaning of the terms. Interacting with computers through QASs, dialogue systems and chatbots is expected to take over searching on the web because of conversational artificial intelligence and great potential and commercial values [187]; dialogue systems and chatbots will reduce the number of questions needed to reach answers.

Recent advances in machine learning and natural language processing will aid search on the web to improve different aspects of the interaction-based search services (i.e., QASs, dialogue systems and chatbots) and understand different ways users naturally interact and interpret information. In consequence, search on the web will manage multiple varieties of interaction between users and search services and users will be able to express their queries in different ways, including text, image, audio, video, or any combination of them. For example, a single search query can be expressed as a combination of text and image [188].

Furthermore, multilingual search will be improved in order to return result for a search query which is expressed in a language different from languages of underlying resources. Search on the web will become more personal and move from returning content from resources towards seeking resolution for users' problems [189]. They may be capable of actively predicting user questions before searching them, providing the users with results they probably need before they truly need and search them [190].

So contextual understanding and personalization for the users would definitely be from the main challenges [190]. Recall the example of food order, users currently take advantage of search services to search and order their foods while it can be imagined that foods will search users in the future, employing the web and proactively, basing on the nutritional condition, context and preferences of the user. With levels of personalization and contextualization, the search will become more conversational and easier in the future.

With regarding such visions of the future, today's search services are still a simple tool, and definitely, there are still many opportunities to make much progress and improve [191]. There are also many challenges to handle, for example, ensuring that the ongoing developments have a positive impact on society as a whole, mitigating negative impacts and ensuring positive impacts for everyone from the big data technology adoption [192]. This is applicable also to the search on the web as an important activity most of us execute on a daily basis.

2.10 Discussion

Throughout this research study, we provide a generic analysis of different services to find information on the web as one of the primary information resources. We start with categorizing services that have been introduced or developed to manage the growing amount of information and data accessible on the web. Here, the search services are systematically summarized and categorized into five groups, including keyword search engines, semantic search engines, QASs, dialogue systems and chatbots. Table 2.1 summarizes the evolution process. Then, we give an overview of the state-of-the-art around search services and highlight the relevant aspects of the services.

To compare the introduced search services, Tables 2.2–2.7 provide

their details on the timeline, approaches and tools, typical usage, pros, cons and open challenges in each search service, respectively.

Table 2.1: Evolution among different search services.

From	To	Description
Keyword search engine	Semantic Search Engine	Keyword search engines look for literal matches of the query words, while semantic search engines not only return results based on keywords but also consider the contextual meaning and the user's intent when fetching the relevant results.
Semantic search engine	Question answering	Semantic search engines fetch relevant results based on the user's query, while QASs allow the user to submit a question and derive the pertinent and exact answers.
Question answering	Dialogue system	The objective of a QAS is to focus on producing the exact answer for a single-user question, while unintentionally ignoring the reasons that motivated that user to pose this question. So, dialogue systems as conversational agents are developed to extend conversations between the user and agent.
Dialogue system	Chatbot	Chatbots are a sub-type of dialogue systems that perform chit-chat with the user or serve as an assistant via conversations.

2.11 Conclusions

This paper overviews different types of search possibilities on the web that help users meet information needs. From the beginning of the web to the present day, keyword-based search engines, semantic-based search engines, QA systems, dialogue systems, and chatbots have been proposed to explore the web. Keyword-based search engines have been the

Table 2.2: Timeline.

Keyword Search	**Semantic Search**	**Question Answering**	**Dialogue System**	**Chatbots**
The first keyword search engine called "Archie" was launched in 1990 [88]. For almost two decades, it was the most common approach that supported browsing services on the web [88]. Keyword search engines were gradually replaced by semantic search engines in the early 10's [193].	Semantic search has emerged with the semantic web in the early 00's [109]. However, the major breakthrough of the semantic search engines as a service was in 2013, when Google presented the "Hummingbird" update [193]. As of today, browsing services supported by semantic search engines play a crucial role in the big picture of the web as a service.	The first QAS was built in 1961 to answer simple questions relating to American League baseball games [2]. With emerging knowledge graphs in 2012, QASs across knowledge graphs have been introduced. Since the early stage, QASs have advanced significantly, particularly about a decade ago, due to progress in natural language understanding and deep neural networks	The concept of conversing in language shifted from fiction to scientific pursuit when Alan Turing forecast machines that could converse like humans in 1949 [194]. Dialogue systems took significant steps in 1966 and 2011 with the emerging ELIZA and Siri, respectively, [152]. Work on dialogue systems has been of broad significance in the recent decade.	In 1966, ELIZA, the first chatbot, appeared intending to imitate a psychologist [195]. PARRY in 1972 was another famous chatbot 16 years later [171]. The first chatbots were relatively simple, and nowadays, they are much more intelligent. Examples of intelligent digital assistants (as they are called nowadays) are Amazon Alexa, Apple Siri, Google Assistant, and Microsoft Cortana [172].

Table 2.3: Approaches/Tools.

Keyword Search	**Semantic Search**	**Question Answering**	**Dialogue System**	**Chatbots**
• Web content mining. • Web usage mining. • Web structure mining.	• Transform unstructured and semi-structured data into structured knowledge. • Schema alignment, entity matching, and entity fusion • Entity, relationship, and document ranking. • Content negotiation to interact with structured knowledge.	• Traditional techniques. • Information retrieval-based techniques. • Semantic parsing-based techniques.	• Pipeline methods. • End-to-end methods. • Retrieval-based methods. • Neural generative methods. • Hybrid methods.	• Parsing. • Pattern matching. • Ontologies. • Markov chain models. • Neural network models.

Table 2.4: Typical usage.

Keyword Search	**Semantic Search**	**Question Answering**	**Dialogue System**	**Chatbots**
For users typing (e.g., "Restaurants" and "Innsbruck"), Keyword Search Engines return web pages containing these keywords. The users have to check the returned pages one by one to find relevant information for them.	For the case of "Pizza", a semantic search engine might return a Pizza description, which details its definition, name, nutritional facts, and local where one can order it.	For users' questions (e.g., "Which restaurants in Innsbruck serve Austrian cuisine on Sundays for between €30 and €60?"), QASs provide precise answers.	Users' intents are identified through a dialogue (e.g., to book a restaurant's table, a dialogue is shaped to understand the user's information needs such as kind of food, location, price range, etc).	For the case of booking a table in a restaurant, users' personal information is used to return a restaurant that matches the user's preferences and there is no need for the users to leave the chatbot and call the restaurant.

Table 2.5: Pros.

Keyword Search	**Semantic Search**	**Question Answering**	**Dialogue System**	**Chatbots**
• Pave the way for the next generation of search engines.	• Improve search accuracy by consuming machine-readable content from websites and structured knowledge from knowledge bases.	• Submit questions in a natural language form. • Supply direct answers to questions.	• Context-awareness due to responding based on the current message and the conversation history. • Response coherence due to maintaining logic and consistency in a dialogue. • Interactive training due to improving themselves via interactions with users.	• Communicate in natural language. • Include personal user information for better search results • Help users in achieving a certain goal (e.g., booking a table in a restaurant) in goal-oriented chatbots. • Interact with external services (e.g., table reservation, hotel room booking).

Table 2.6: Cons.

Keyword Search	**Semantic Search**	**Question Answering**	**Dialogue System**	**Chatbots**
• False positive due to polysemy words. • False negative due to synonym words. • Be literal and ignore the context of words as well as the intentions of users. • Ineffective information retrieval process in large-scaled resources.	• Complex implementation. • Limited powerfulness due to the knowledge acquisition process. • Rely on the quality of knowledge sources.	• Lack of conversation to verify users' intents in complex questions. • Lack of dialogues to fulfill all aspects of information needs once users' requirements can not be expressed using a single question and requires the user to ask multiple questions.	• Privacy threats due to serving many users, having the ability to learn through interactions and storing sensitive information.	• Lots of manual work in formulating training phrases. • Low accuracy for neural network models with no predefined training phrases. • Low response diversity.

Table 2.7: Open challenges.

Keyword Search	**Semantic Search**	**Question Answering**	**Dialogue System**	**Chatbots**
• No significant open challenge.	• Build more scalable semantic search engines according to the growth of the semantic web. • Curate and update knowledge bases to offer high-quality knowledge due to the heavy dependence of semantic search engines on the knowledge bases to deliver information.	• Understand complex semantics in input questions. • Reduce huge search space in knowledge bases. • Incomplete knowledge bases.	• Incremental processing and swift warm-up. • Perceive and express emotions. • Privacy protection.	• Context awareness. • Diversity of responses (natural language generation). • Chatbots with a stronger personality.

first generation of search engines, and then semantic search engines have emerged to solve the limitations of the keyword-based search engine. QASs, dialogue systems (as a subdivision of QASs) and chatbots (as a subdivision of dialogue systems) are targeting to assist users find pertinent information through a natural language interaction. We analyze the current state and related works, process model, data preparation and representation, methodologies, and categories for these types of search services in our research study. We also show that the development of the search on the web, however, is still an ongoing process. The future systems will focus on multiple varieties of interaction and multilingual search. Those systems are expected to deliver answers to users even before getting the questions asked by the users however such and similar developments comprise societal, business and ethical issues that need to be considered when designing the search on the web of the future.

To the best of our knowledge, this survey is the first, most comprehensive and up-to-date one currently covering all the web search possibilities throughout the years. This work helps academics and developers who need to grasp all the search services quickly and understand their developments and evolution and also provides a good starting point for them to probe deeper into each of these services.

We also acknowledge that our study can not analyze all aspects of search services with technical discussions. A future research direction is to evaluate and compare the efficiency and accuracy of the retrieval results of the services [72]. Another possible direction for future work is to focus on a relatively small time span (e.g., the latest ten years) and deepen the discussions and analysis of the literature.

Chapter 3

Question Answering over Knowledge Graphs: A Case Study in Tourism

Contents

3.1 Abstract

Over the recent years, a large number of knowledge graphs (KGs) have been developed to store and present small and medium enterprises' data in the form of subject-predicate-object triples. The KGs are not easily accessible to end-users because they need an understanding of query languages and KGs' structures. To assist end-users in accessing these KGs, question answering over KGs targets to provide answers for natural language questions (NLQs). This paper proposes an approach to answer questions over small and medium scaled KGs based on graph isomorphism in two phases: (1) offline phase and (2) semantic parsing phase. In the offline phase, a semi-automated solution is proposed to generate NLQs and their answers, which are used to train machine learning models employed in the next phase. In the semantic parsing phase, a given input NLQ is mapped into a query pattern according to its grammatical structure. Each query pattern contains some slots that need to be filled with corresponding entities, classes, and relations from the KG. While string and semantic similarity metrics are applied to identify the entities and classes, the probability distribution of the relations is used to extract the relations. The Cartesian product of the identified entities, classes and relations is utilized to fill the slots, derive SPARQL queries, and finally retrieve the answers. To evaluate the proposed approach, we use SalzburgerLand KG, a real KG describing touristic entities of the region of Salzburg, Austria. Our results show that the approach improves the end-to-end user experience in terms of interactive question answering and performance.

Keywords: Knowledge Graphs, Question Answering, Semantic Parsing, Small and Medium Enterprises

Publication: Sareh Aghaei, Elie Raad, and Anna Fensel. "Question Answering Over Knowledge Graphs: A Case Study in Tourism". In: IEEE Access

Vol. 10 (2022), pp. 69788–69801. DOI: 10.1109/ACCESS.2022.3187178

3.2 Introduction

With the increasing growth of web data, a large number of knowledge graphs (KGs) have become available on the web. A KG is a structured representation of real-world entities which are connected by semantically-interrelated relations [196, 197]. A Resource Description Framework (RDF) KG [198], which can also be viewed as a labelled graph, is a collection of RDF triples including three fields: subject, predicate and object [199].

Beyond the generic, huge, and open-world KGs with millions or billions of facts (e.g., DBPedia [22]), KGs are increasingly used in business scenarios [200]. Most of the recent KGs are domain-specific, with thousands or hundreds of thousands of facts in various domains such as tourism, healthcare, and manufacturing [40, 201]. The last few years have seen a spike in building KGs by Small and Medium Enterprises (SMEs). For example, SMEs in the manufacturing and production industry utilize KGs to organize data generated by different machines during manufacturing processes [43]. In the tourism domain, websites use small and medium scaled KGs to manage tourism information for different regions and improve the traveller experience [202, 203, 204, 205, 206].

Query languages such as SPARQL[1] are used to access knowledge stored in RDF KGs. Since writing queries can become quite tedious and challenging for end-users, question answering (QA) systems have been introduced to simplify access to the KGs. These systems allow end-users to take advantage of semantic web standards' expressive power while simultaneously hiding their complexity behind intuitive and easy-to-use

[1]Simple Protocol and RDF Query Language

systems [207, 208]. Therefore, a KG-based question answering system (KGQAS) can be defined as an easy-to-use system that aims at making the facts of KGs accessible and beneficial for end-users [207].

Natural language questions (NLQs) in KGQASs are categorized based on different criteria. According to question types formulated by end-users, questions are classified into three major groups, including (1) factoid questions, (2) confirmation questions, and (3) hypothetical questions [209]. The factoid questions are factual in nature and commonly start with a WH-word[2]. The confirmation questions are generally answered in the form of "yes" or "no" through verification and justification. The hypothetical questions have no specific correct answers, and the answers are subjective. The number of hops required to reason over KGs to obtain answers is another criterion that divides NLQs into two categories: simple questions and complex questions. A simple question, namely a single-hop question, is answered through only one hop, whereas a complex question, called a multi-hop question, requires reasoning over two or more hops of the KG [10, 24, 210]. The target of this paper is factoid questions, both simple and complex.

Although many KGQASs have been proposed in recent years, they often require a lot of training data, which is usually unavailable in small and medium scaled KGs (including facts ranging from thousands to hundreds of thousands). The same observation is made in our project WordLiftNG [17] when working with SMEs. In the project, KGs are created for SMEs' websites (e.g., tourism websites), while to make these KGs accessible for users using machine learning techniques, training examples are hardly available. We conduct a literature review (see Section 3.3) and observe that the primary attention has been paid to huge KGs, which is why we aim to propose an approach to answer questions over small and medium

[2]when, where, who, what, etc

scaled KGs where there is little or no training data.

This paper tackles three challenges in providing KGQASs for SMEs as follows.

1. Lack of training data is a primary challenge in small and medium scaled KGs to develop KGQASs. Despite numerous possibilities to use machine learning ideas to develop KGQASs, those ideas cannot be accomplished on this scale without relevant data.

2. Traditional KGQASs rely on hand-crafted templates, which require a lot of effort to cover every possible question and cannot be easily adapted to other KGs.

3. The linguistic gap between questions and KGs' vocabularies can lead to poor performance in answering questions, particularly multi-hop questions that generally include more complex semantic information than simple questions.

Due to the aforementioned challenges, the main research question of the study is *how to leverage machine learning and semantic web to answer questions asked by end-users using facts stored in small and medium scaled KGs?*. The research sub-questions derived from the main research question can be summarized as (1) how to alleviate the training data problem in QA over small and medium scaled KGs? (2) how to overcome the task of defining templates for each possible question?, and (3) how to bridge the linguistic gap between question sentences and KGs' vocabularies?

According to the described challenges and research sub-questions, the main contributions of this paper can be summarized as follows:

1. A semi-automatic generic approach is introduced to create training data using facts stored in KGs.

2. NLQs are automatically mapped into query patterns according to questions' grammatical information.

3. Different similarity measurements are employed to identify entities and classes of a given input question, and also a multi-label classifier is applied to extract relations.

4. A real KG describing touristic entities is used to show how the proposed approach can be applied in practical use-cases. Additionally, we show how the approach improves the end-to-end user experience in terms of accuracy, recall, precision and F1-score.

In the light of industrial requirements and current challenges in SMEs to develop KGQASs, this paper is built around a practical use-case in tourism from the project WordLiftNG. Here, we target to enable end-users to ask their information needs through factoid questions and get their answers. For example, the question "Which hotels offer pet-friendly rooms in Salzburg?" is posed by a user, and then the answer is returned.

In this research study, we present a graph isomorphism-based approach for QA over small and medium scaled KGs consisting of two phases: an offline phase followed by a semantic parsing phase. The offline phase introduces a solution to generate training data from facts stored in a KG. Then, the generated training data is used to build a semantic parser to answer a given input question in the semantic parsing phase.

The remainder of the paper is organized as follows. Section 3.3 reviews the related works, Section 3.4 formulates the problem, Section 3.5 introduces the proposed approach, Section 3.6 presents the experiment and evaluation results, Section 3.7 provides the discussion and Section 3.8 contains the conclusion and future work.

3.3 Related Works

This section summarises the related work on QA over KGs. Since transforming a set of RDF triples into informative text is a task of natural language generation, and the given focus of our study is to provide a KGQAS for SMEs, we restrict the literature review to the research progress in the area of QA over KGs.

Generally, related works in QA over KGs can be divided into three main groups, namely, template-based techniques, information retrieval-based techniques and semantic parsing-based techniques. We briefly review each of these three groups as follows.

1. Template-based techniques: Template-based KGQASs take advantage of templates or rules to answer questions through mapping questions to predefined templates [4, 14, 211, 212]. Although these approaches lead to high precision, recall is low due to low coverage of the variety of questions [213]. Additionally, adaptation for various domains is difficult in these approaches. In the context of the movies and cinemas domain, an ontology-based system has been introduced in [214]. This system uses a set of questions called predictive questions, which are likely to be asked by users in the domain ontology. Then, a corresponding query template is generated according to each predictive question that can be used only to extract the answer to that question from the KG. The presented model in [215], namely Aqqu, maps questions to three templates that basically have limited coverage on complex questions. Then, all entities that match a part of the question are identified from the KG. Next, Aqqu instantiates the three templates and chooses the best instantiation based on a ranking model to query the KG and return the answers.

2. Information retrieval-based techniques: These approaches focus on retrieving all candidate answers and then ranking them to select the best answers instead of parsing the NLQ to obtain a formal semantic representation. The state-of-the-art works [7, 8, 9, 216] leverage neural networks to generate distributed representations of questions and candidate answers. These networks are trained with more than 4K questions paired with answers which are rarely available in the case of small and medium scaled KGs. For example, EmbedKGQA [9] uses KG embeddings to answer multi-hop NLQs. First, it learns a representation of the KG in an embedding space using ComplEx embedding. It then learns a question embedding using a feed-forward neural network for a given question. Finally, it combines these embeddings to predict the answer through a scoring function.

3. Semantic parsing-based techniques: The semantic parsing-based techniques conceptualize the task of QA over KGs to parse NLQs and then convert the questions to logical forms or structured queries such as SPARQL queries. Neural semantic parsing approaches can cover more complex questions. However, it is challenging to train a neural semantic parser due to the lack of a considerable amount of gold logical forms [10]. A structure, namely the syntactic graph, is introduced in [12] to represent the intention of a given input question using three types of syntactic information, including word order, dependency and constituency features. Then a graph-to-sequence model is employed to encode the syntactic graph and decode a logical form for the question. A recursive neural network-based approach is introduced in [217] to learn and classify questions into their corresponding query patterns using a Tree-Long Short Term Memory (LSTM) model. This model is trained over

LC-QuAD dataset[218] which includes 5K questions paired with their SPARQL queries. A comparative study is presented in [219] using the LC-QuAD 2.0 dataset [220] that consists of 30K question-answer pairs. The objective of this study is to compare different classifiers, including random forest classifier [221] and XGBoost classifier, based on different pre-processing techniques, including POS tags, word embeddings and combination of POS tags and word embeddings. Similar to [217, 219], TeBaQA [222] leverages question classification to shift the QA problem into a classification task. In TeBaQA, syntactic and semantic features are used to train a statistical classifier. TeBaQA categorizes questions based on their subject areas (e.g., film, music, or city) to calculate semantic features. While some information, such as the number of verbs or adjectives, is used as syntactic features, dependencies of questions' words do not contribute to shaping feature vectors. Note that in semantic parsing-based approaches, the DBPedia-based annotator tools (e.g., DBpedia Spotlight [223]) or DBPedia-based lexicons (e.g., RNLIWOD[3]) are mainly applied to address the tasks of entity linking or relation extraction, which are limited to DBPedia KG [217].

Although template-based approaches can be applied in small and medium scaled KGs, they require much effort to ensure covering every possible question and cannot be easily adapted to new domains. Different from the information retrieval-based and semantic parsing-based methods in [219, 217] which require a considerable amount of training data, we propose an approach to automatically classify questions into their corresponding query patterns for SMEs. First, we deal with the lack of training data in small and medium scaled KGs by generating NLQs over

[3]NLIWOD - Natural Language Interfaces for the Web of Data: `https://github.com/semantic-systems/NLIWOD`

RDF triples in a semi-automated solution. Then, in contrast to [222], our approach analyzes questions based on purely syntactic features, including POS tagging and dependency parsing. While the state-of-the-art approaches widely apply DBPedia-based annotators and dictionaries to identify entities and relations, we define a similarity score and a predicate classifier in our work. String and semantic similarity scores of n-gram words collected from the NLQ and KG's entities and classes are used to define the similarity scores and identify the entities and classes mentioned in the NLQ. Also, the predicate classifier calculates the probability distribution of relations and finds the relations with the highest probability.

3.4 Problem Definition

Formally, we denote[4] a KG as $KG = (N, E, F)$, where N and E are the sets of entities (i.e. nodes) and relations (i.e., edges), respectively and F is a set of facts. Each fact refers to a triple (s, r, o) to present relation $r \in E$ between the subject $s \in N$ and object $o \in N \cup C$, where C is a set of the used classes[5] in the KG. Once the number of facts $|F|$ is around thousands or hundreds of thousands, the KG's scale is assumed to be small and medium.

Given an available small and medium KG $KG = (N, E, F)$ and a NLQ q in the format of a sequence of tokens, the task of QA over KGs aims to answer the question q using the facts F stored in the KG.

[4]A KG can be defined as a combination of instance data (ABox) and ontology data (TBox). Since the ontology is not considered in this study, we limit our definition to KGs containing instance data.

[5]In RDF, anything with a subject Uniform Resource Identifier (URI) is called a resource and a class is simply a way of defining groups into which resources can be meaningfully placed.

3.5 Proposed Approach

In this section, our approach is presented. An overview of our approach is given in Section 3.5.1, and the details of the approach are provided further. Section 3.5.2 presents the details on how questions are generated over RDF triples in the pre-processing, offline phase. Further, Section 3.5.3 lists and discusses all the steps of the semantic parsing phase.

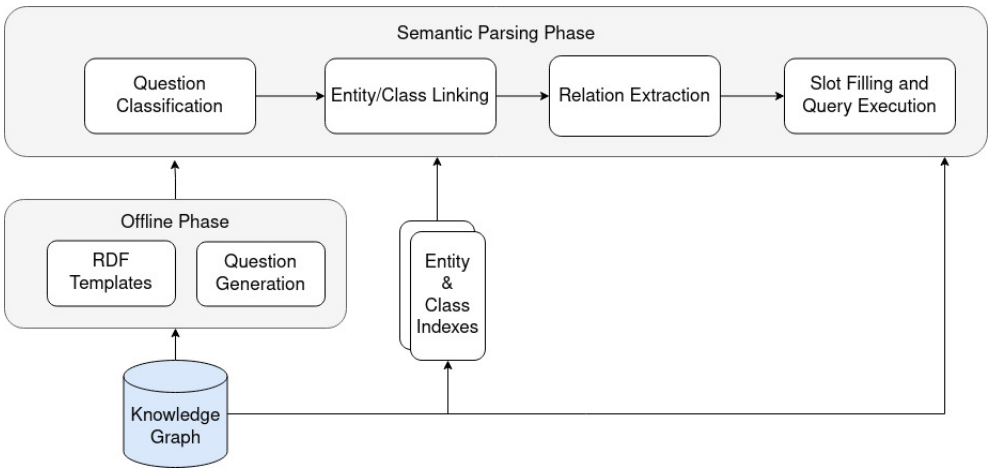


Figure 3.1: An overview of the proposed approach.

3.5.1 Overview

In our work, the proposed approach consists of two phases, including an offline phase and a semantic parsing phase, as shown in Figure 3.1. The offline phase applies a semi-automated solution to generate training data containing question and answer pairs, query patterns and relations. The semantic parsing phase consists of four steps, including (1) question classification, (2) entity and class linking, (3) relation extraction, and (4) slot filling and query execution. The objective of question classification is to utilize structural information in questions to learn an XGBoost classifier[224] and then assign a query pattern to a given input question. Since SPARQL is basically a graph-based query language, isomorphic SPARQL queries are used to determine query patterns. We conduct part

of speech (POS) tagging and dependency parsing techniques to analyze the structural information of questions. The assigned query pattern contains empty slots, which are then instantiated after the entity and class linking, and relation extraction steps. In the entity and class linking step, we utilize string and semantic similarity metrics to map entities and classes mentioned in the question sentence with corresponding entities and classes in the KG. Additionally, a multi-layer perceptron (MLP) classifier is learned to predict the probability distribution of the KG's relations in the relation extraction step. Finally, the Cartesian product of the possible values in the slots is applied to fill the slots and retrieve the answers.

3.5.2 Offline Phase

We choose a semi-automatic way to generate questions over RDF triples for two main reasons. First, manually generating questions would be too costly, requiring a certain level of knowledge of the underlying KG's domain. Second, automatically generating questions using neural networks or natural language models requires large KGs and training datasets to achieve good performance (these KGs are basically larger than the small and medium scaled KGs) [225, 226, 227].

In this offline phase, we define a set of RDF templates according to the maximum number of hops in complex questions as a first step. In the following step, the RDF templates are utilized to generate NLQs through querying the KG and verbalizing query results. Figure 3.2 depicts the workflow of the offline phase, which is explained in detail as follows.

3.5.2.1 RDF Templates

To define RDF templates, the maximum number of hops n in complex questions needs to be determined. Basically, the number of required hops

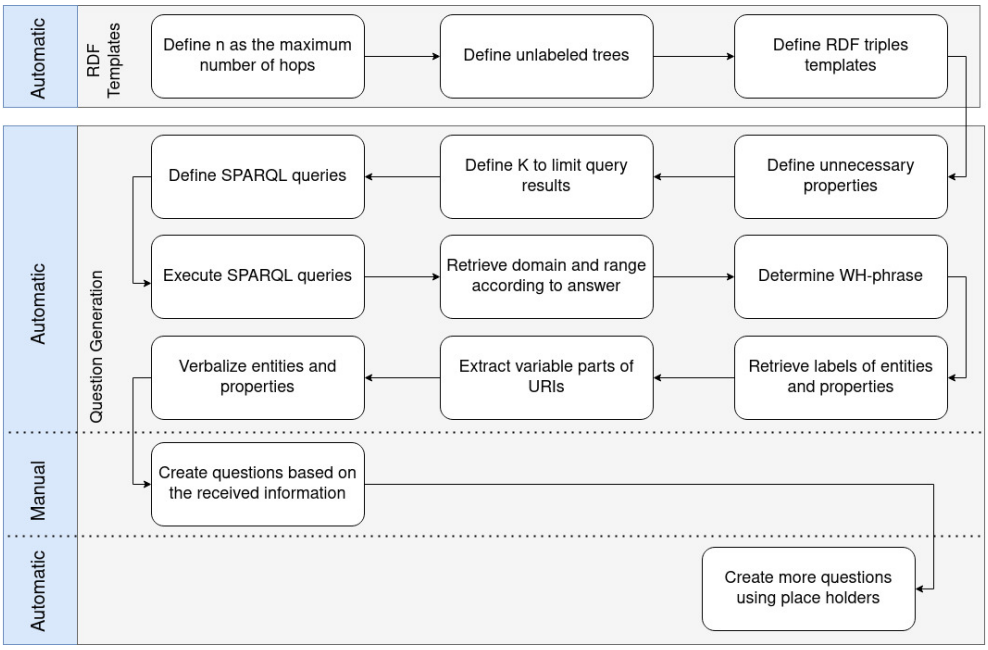


Figure 3.2: The workflow of the offline phase.

for reasoning over triples of a KG to find answers does not exceed 4 in real scenarios. The statistics of the existing benchmark QA datasets [218, 228, 229, 230, 231] confirms this statement. Thus, this paper assumes that the required number of hops is $n <= 4$.

With the assumption of $n <= 4$, all the possible unlabelled trees formed from n hops are defined. An unlabelled tree is assumed to be a tree whose nodes are not explicitly labelled. Thus, we are interested only in tree structures when counting unlabelled trees consisting of n hops. As shown in Table 3.1, the number of unlabelled trees for 1, 2, 3 and 4 hops are 1, 1, 2 and 3, respectively.

Then, each unlabelled tree is presented with a set of RDF templates, where the nodes and edges represent entities (subjects, objects or classes) and properties, respectively. Note that all the possible states to represent an n-hop unlabelled tree using RDF triples is equal to 2^n as shown in Table 3.1. For example, the RDF templates of the unlabelled tree with $n = 2$ are depicted in Figure 3.3.

Table 3.1: Statistics of unlabelled trees and RDF template based on hops.

Hop	Unlabelled tree	RDF template
1	1	1×2
2	1	1×4
3	2	2×8
4	3	3×16

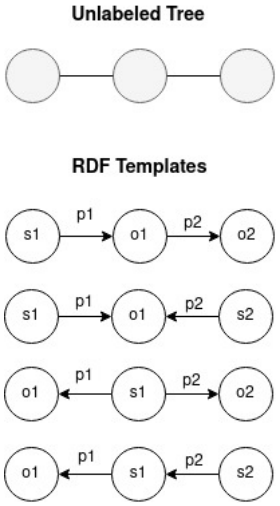


Figure 3.3: Unlabelled tree and RDF templates with 2 hops.

3.5.2.2 Question Generation

According to each RDF template, a SPARQL query is defined to retrieve those entities and properties of the KG mapped to the RDF template. The defined SPARQL query includes a condition to filter out the unnecessary properties and a solution modifier to limit the number of returned rows to k.

Generally, some properties in a KG are unnecessary in NLQ generation, such as the properties used to link the KG entities to the entities in other KGs (e.g., dbo:sameAs, dbo:seeAlso). We filter out all the unnecessary properties from the SPARQL queries to prune the irrelevant parts of the KG.

Moreover, to get random results from a SPARQL query, the SPARQL's built-in RAND() function is used to order results by that random number. The idea behind retrieving k random rows is to shuffle the facts' KG and then have an approximate balance between the distri-

bution of templates based on the number of hops (e.g., the 1-hop RDF template can retrieve all the facts of the KG).

Therefore, given a RDF template, a SPARQL query is executed, and then a verbalization process is performed over the query results. We use the labels of entities (non-literal) and properties to verbalize them. If an entity or edge does not have a label in the KG, the variable part of its URI is adopted (e.g., we use "farmhouse holidays" as the variable part of the URI <http://open.salzburgerland.com/en/entity/farmhouse_holidays>). Additionally, an entity is randomly selected as the answer entity for each row query result. Then, the domain and range of the property connected to the answer entity are fetched to determine the WH-word of the factoid question (e.g., once the domain is "Place", the relevant WH-word is "Where").

Thus, the information obtained for each row query result includes the verbalized triples, the answer and the WH-word. We employ this information to create a question manually in the next step. Since humans create questions, they are expressed differently due to humans creating questions based on their own vocabularies. For example, to make a simple question for the information "(Mozart Week, startDate, 22 January), answer: Mozart Week, WH-word: When", different questions may be created such as "When does Mozart Week start?", "When does Mozart Week begin?" and "What is the starting date of Mozart Week?".

Next, we utilize each obtained question as a pre-defined question sentence q with placeholder (entity holder, class holder and relation holder) variables (e.g., "When does [entity_holder] [begin]") to create more questions automatically through running their corresponding SPARQL queries. Thus, the placeholders of the question sentence q are replaced with the verbalized forms of the retrieved values to shape new NLQs.

For each generated question, the information, including the answer,

the RDF template, the number of hops, the required number of entities, classes and relations as the placeholders (to make a SPARQL query) and also the corresponding KG's relations are collected to be fulfilled in the semantic parsing phase. As an example, according to Figure 3.4, for the question sentence "What is the address of the hotel where Mozart Week takes place" the numbers of hops, entity holders, class holders, and relation holders are 3, 1, 1, and 2 respectively and the corresponding relations include <http://schema.org/address> and <http://schema.org/organizer>.

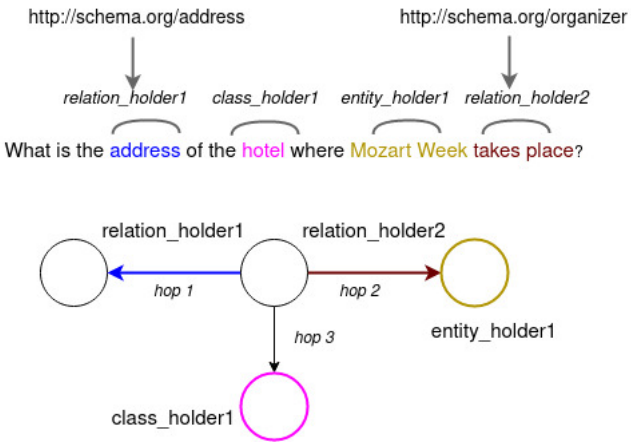


Figure 3.4: An example of collected information of a question.

3.5.3 Semantic Parsing Phase

Query patterns as a significant part of QA over KGs simplify semantic parsing of input questions and creation of structured queries to retrieve answers [215]. Since SPARQL is basically a graph-based query language, an isomorphism can be used to determine the structural equivalence of two SPARQL queries [222]. According to graph isomorphism, two query patterns, which basically are labelled graphs, are isomorphic if there is an edge-preserving node bijection[6] between entity (and class) sets.

[6]A bijection is a bijective function that establishes a one-to-one correspondence between elements of two given sets. Here, sets are considered as the sets of graphs' nodes.

For example, although questions "What is the address of the hotel where Mozart Week takes place?" and "What is the phone number of the ski-resorts that open in October?" are semantically different, their query patterns include the same number of nodes and edges and also their edge connectivity is retained. Thus, these questions are answered through one query pattern due to the structural similarity of their SPARQL queries, as shown in Figure 3.5.

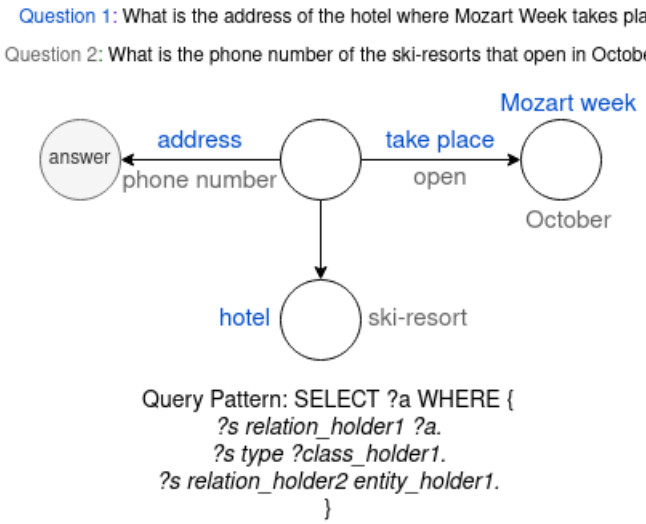


Figure 3.5: Semantically different questions with similar query pattern.

Thus, we determine a set of query patterns as question classes using graph isomorphisms according to the obtained information in the offline phase (including varied questions and their RDF templates and the number of entity, class and relation holders). The graph patterns comprise some slots (entity holders, relation holders and relation holders) independent of the KG's domain and vocabulary. In section 3.6.2, we show all the basic query patterns used as the question classes in our use-case.

3.5.3.1 Question Classification

After determining the query patterns, a classification model is employed to classify NLQs into their corresponding query patterns. We use XGBoost classifier due to it is a boosting classifier, which combines tree mod-

els with lower classification accuracy and builds a highly accurate and low false positive model through the constant iteration of the model[224].

XGBoost (Extreme Gradient Boost) classifier is considered an implementation of gradient boosted decision trees (GBDTs) for classification. A GBDT is a decision tree ensemble learning algorithm that combines multiple machine learning models to produce improved results using a gradient descent algorithm. Gradient boosting is based on the idea that combining the best possible next model with the prior models minimizes overall prediction errors.

Thus, an XGBoost Classifier is trained to assign an appropriate query pattern to a given input question. The features are calculated based on the grammatical structure of questions, including POS tagging and dependency parsing. Also, we use the padding and label encoding to accomplish inputs with the same size in numeric forms, respectively.

Part of Speech (POS) Tagging: POS tagging, a very basic and well known natural language processing problem, is used to classify words of a sentence into their corresponding part of a speech (verb, noun, adjective, etc) and label them accordingly. This paper applies the averaged perceptron tagger from the Natural Language Toolkit (NLTK) package[7] which is based on the average perceptron machine learning algorithm, to generate POS tags of the question sentences.

Dependency Parsing: Dependency parsing is used to analyze the grammatical structure of a sentence according to the dependencies between the words of the sentence. Dependency triples represent the relationships between two words (headword and dependent word) using dependency tags. Since the output from the parse tree needs to be vectorized, we conduct dependency triples of a question sentence in the form of <POS tag of headword, dependency tag, POS tag of dependent

[7] `https://www.nltk.org/`

word>. Considering the question sentence "When does Mozart Week start?", Figure 3.6 illustrates the dependencies among the words. For example, there is the dependency aux (i.e., auxiliary) from "start" with POS tag VB (i.e., verb, base form) to "does" with POS tag VBZ (i.e., verb, 3rd person sing. present) that is considered as a triple <VB, aux, VBZ>.

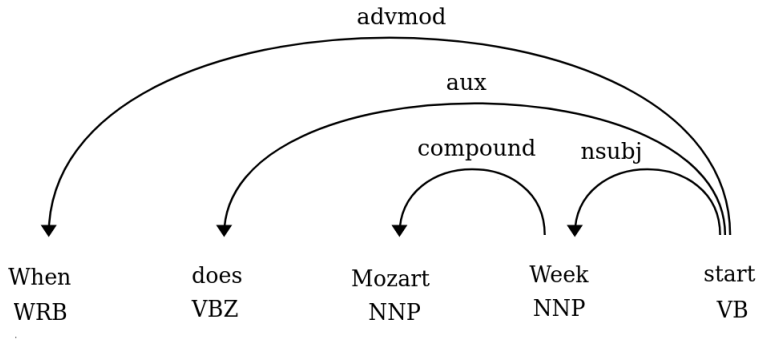


Figure 3.6: An example of dependency parsing.

Padding: Padding is commonly used in natural language processing tasks to obtain feature vectors with the same length. This paper uses post-padding to pad feature vectors to a preferred length.

Label Encoding: Basically, machine learning algorithms such as XGBoost will perform better in terms of accuracy and other performance metrics when the data is expressed as a number (machine-readable form) instead of categorical. Here, label encoding converts the POS tags and dependency triples into numeric forms. All the possible POS tags (in default tagger of NLTK package) and dependency tags are considered to encode features, which are 37 and 63, respectively. So, the maximum number of possible triples in the form of <POS tag, dependency tag, POS tag> is equal to $37 \times 63 \times 37 - 37$ that we consider all the possible triples though some triples may never happen.

Thus, given a question, the POS tags and dependency triples are generated and padded with the maximum lengths l_1 and l_2 (determined

according to maximum values in training data), respectively. Next, the padded vectors are encoded and then concatenated to shape the feature vector of the question with length $l_1 + l_2$. Once the XGBoost classifier is trained, it is able to predict the question's query pattern.

Figure 3.7 shows the whole process of POS tagging, dependency triples, padding and encoding of the example question sentence "What is the address of the hotel where Mozart Week takes place?". In this example, we use "x" and "y" to pad the POS tags and dependency triples, respectively.

3.5.3.2 Entity Linking and Class Linking

The task of entity linking can be defined as linking entities from the KG that are mentioned in the question sentence. To end this, our introduced technique includes three stages: (1) n-gram collection, (2) candidate retrieval from the KG, and (3) entity selection. Here, we use the same technique to link class mentions appearing in questions with their corresponding classes in the KG.

First, word n-grams are collected (as the entity or class mentions) using a fixed-sized sliding window that runs from start to end of a question sentence (an n-gram is a subsequence of length n from an item sequence). Here, we apply unigram, bigram and trigram to extract all possible tokens. Taking the word sequence in the question sentence: "When does Mozart Week start?" as an example, there are five 1-gram (i.e., unigram): "When", "does", "Mozart", "Week" and "start", four 2-grams (i.e., bigrams): "When does", "does Mozart", "Mozart Week", and "Week start", and three 3-grams (i.e., trigrams): "When does Mozart", "does Mozart Week", and "Mozart Week start". In the stage of candidate retrieval, two indexes are created based on labels of entities[8] and classes,

[8]Either URI resources or literal resources

Question: What is the address of the hotel where Mozart Week takes place?

POS tags: ['wp', 'vbz', 'dt', 'nn', 'in', 'dt', 'nn', 'wrb', 'nnp', 'nnp', 'vbz', 'nn']

Dependency triples: ['wp cop vbz', 'wp nsubj nn', 'nn det dt', 'nn nmod nn', 'nn case in', 'nn det dt','nn acl:relcl vbz', 'vbz advmod wrb', 'vbz nsubj nnp', 'nnp compound nnp', 'vbz obj nn']

Padding: ['wp', 'vbz', 'dt', 'nn', 'in', 'dt', 'nn', 'wrb', 'nnp', 'nnp', 'vbz', 'nn', x, x, ..., x]

Padding: ['wp cop vbz', 'wp nsubj nn', 'nn det dt', 'nn nmod nn', 'nn case in', 'nn det dt', 'nn acl:relcl vbz', 'vbz advmod wrb', 'vbz nsubj nnp', 'nnp compound nnp', 'vbz obj nn', y, y, ..., y]

Feature vector: [36, 34, 4, 14, 8, 4, 14, 38, 15, 15, 34, 14, 1, 1, .., 1, 83456, 84099, 32960, 33321,32613, 32960, 32288, 78429, 79498, 35038, 79692, 4602, 4602, 4602, 4602, ..., 4602]

Figure 3.7: An example of feature vectors in the question classifier.

respectively (if an entity or class does not have a label, the variable part of its URI is adopted). To select relevant entities and classes in the last stage, we leverage Levenshtein edit distance[232] to compute string similarity as well as word embeddings in order to create embeddings of mentions and candidates. This enables us to calculate the semantic score using cosine similarity.

Levenshtein edit distance is computed through a dynamic programming algorithm that addresses the problem of string matching based on various edit operations, including substitution, deletion or insertion [233]. Here, the minimum number of single-character insertions, deletions, and substitutions required to transform a mention into a candidate is considered as their Levenshtein edit distance. We consider the length of the candidates in finding their similarities (lines 2 and 3 of Algorithm 1) to prioritise the longest word combinations rather than the words that make it up. To accomplish word embeddings, we apply the pre-trained model SBERT[9] to generate the embeddings of the mentions and candidates and then compute the cosine similarity between the embeddings. While BERT[10] is a state-of-the-art pre-trained contextual language representation model built on a multi-layer bidirectional transformer encoder [62], SBERT is a modification of the pre-trained BERT network to derive a semantically meaningful word or sentence embeddings using siamese and triplet network structures [234]. Thus, given each entity and class mention collected by n-grams, we calculate their similarity scores with candidates from entity and class indexes according to the pseudo-code shown in Algorithm 1.

Next, the entities and classes with the highest similarity scores are selected according to the required number of entity holders and class holders in the query pattern.

[9]Sentence-Bidirectional Encoder Representations from Transformers
[10]Bidirectional Encoder Representations from Transformers

Algorithm 1 Similarly Score Computation

1: Compute vector STS (STring Similarity) containing the Levenshtein edit distance ld between entity mention m and entity candidates
2: Sort STS based on ld/ls ascending where ls is the length of the entity candidate
3: Sort STS based on ls descending if ld is equal to zero
4: Compute STR (STring Ranks) including the entity candidates' string-ranks where the string-rank of the entity candidate c is $1/index(stc)$ where $index(stc)$ is is the index of the entity candidate c in STS vector
5: Compute vector SES (SEmantic Similarity) containing the cosine similarity cs between the embeddings of an entity mention m and the embeddings of entity candidates
6: Sort SES based on cs descending
7: Compute SER (SEmantic Ranks) including the entity candidates' semantic-ranks where the semantic-rank of the entity candidate c is $1/index(sec)$ where $index(sec)$ is is the index of the entity candidate c in SES vector
8: Compute similarly score of an entity mention m and the entity candidate c based on the sum of their ranks in STR and SER

3.5.3.3 Relation Extraction

The task of relation extraction targets at finding the specific predicates (also named properties or relations) from the KG that match the phrases detected in a given question sentence. Basically, the task of relation extraction is more difficult than the detection of entities and classes due to the large number of expressions that can be used to express the same predicate [211]. Since a question sentence can include more than a relation (such as the described questions in Figure 3.5), we utilize multi-label classification. The task of multi-label classification is concerned with learning from a set of samples that are associated with a set of labels. So, zero or more labels are required as output for each input sample. To accomplish relation extraction, given a question, probability distributions for predicates are computed by an MLP predicate multi-label classifier. Note that the unnecessary predicates which are identified in the offline phase are not considered in this calculation.

An MLP classifier is a neural network that follows a feed-forward

mechanism and maps input data onto corresponding outputs, and the neurons in the MLP are trained with the back propagation learning algorithm. It consists of three types of layers, including the input layer, output layer and hidden layer. The input layer receives the data to be processed. Next, the hidden layers, which are an arbitrary number of layers between the input and output layer, are the true computational engine of the MLP. Finally, the task of classification is performed by the output layer.

We use the pre-trained model SBERT to generate the embedding of question sentences. Then the embedding is passed through an MLP classifier with two hidden layers (in our use-case) followed by a sigmoid activation function in the output layer. The sigmoid function is the right choice due to it predicts the probability for each class label (a value between 0 and 1), and the predicted probabilities are independent (the probabilities-sum does not need to be 1).

Once the predicate classifier outputs the probability of each predicate in an input question, the top-k relations are selected as the most relevant extracted relations where k is equal to the number of required relation holders in the assigned query pattern.

3.5.3.4 Slot Filling and Query Execution

Once entities, classes and relations that can fill the slots of a query pattern are identified, we arrive at a range of possible states to fill the slots using the Cartesian product of the possible values. Since the maximum number of hops in a query pattern equals 4, the total number of permutations to fill slots is limited. So, a SPARQL query is executed according to each permutation, but in most queries, no results are returned. Finally, the union of all the returned results is considered the answer.

3.6 Experimental Study

This section contains details about the experimental study. In Section 3.6.1, the applied KG in our use-case is introduced. Section 3.6.2 explains the experiment process in the offline and semantic parsing phases, and Section 3.6.3 provides information on the evaluation metrics and the reasons why those evaluation metrics are considered. Finally, in Section 3.6.4, we present our experimental results.

3.6.1 Dataset

With the rapid increase in using KGs to store and present data generated by SMEs, KGs have been developed in tourism websites in recent years. So, we choose a KG in the tourism domain to showcase the proposed approach. The KG chosen for this empirical study is from Salzburgerland[11] (this KG is a part of the project WordLiftNG[12]). It consists of approximately 31K facts describing touristic entities of the region of Salzburg, Austria.

3.6.2 Experiment Process

To construct the training data, the unnecessary properties of the KG are identified in the offline phase. Among the 39 properties that are present in the Salzburgland KG, 10 properties are identified as the unnecessary properties, as tabulated in Table 3.2.

Out of all the possible RDF templates in Table 3.1 (70 RDF templates[13]), we execute SPARQL queries where some of them may return no result. Since the objective is to arrive at an almost balanced distribution among questions based on the number of hops to return the answers,

[11] `http://data.salzburgerland.com/en`

[12] Eurostars funded project WordLiftNG aims to construct the most SEO-friendly structured linked data (e.g., KGs) for SMEs.

[13] $1 \times 2 + 1 \times 4 + 2 \times 8 + 3 \times 16 = 70$

Table 3.2: Unnecessary properties in Salzburgland KG.

property
http://schema.org/description
http://schema.org/image
http://www.w3.org/2002/07/owl#sameAs
http://schema.org/sameAs
http://purl.org/dc/terms/references
http://schema.org/url
http://purl.org/dc/terms/subject
http://schema.org/dateModified
http://schema.org/datePublished
http://schema.org/interactionCount

the values of K (the number of returned results) in RDF templates are different. The obtained information from each row result includes the verbalized triples, the answer entity, and the WH-word question. These elements are used to generate questions by the authors involved in the paper. Once all the questions are generated, we categorize the questions based on the isomorphic graph patterns, the required numbers of entity holders, class holders and relation holders and finally arrive at 7 query patterns (we ignore the patterns with examples less than 5% of the total number of generated questions) that Table 3.3 shows the statistics.

Table 3.3: Statistics of query patterns.

Query pattern	Hops	entity holders	class holders	relation holders	Questions
1	1	1	0	1	98
2	2	1	1	1	103
3	2	2	0	1	106
4	3	1	1	2	102
5	4	1	1	3	78
6	4	2	1	2	117
7	4	2	1	3	151

To present feature vectors of the questions, the maximum numbers of POS tagging and dependency triples of the observed sequences are considered 17 and 18, respectively. We use 'x' and 'y' to pad the POS tags and dependency triples, respectively, to make the samples in the same size (the length of the feature vectors is equal to 35).

We apply the pre-trained model *distilbert-base-nli-stsb-mean-tokens*

with a dimension of 768 to generate the feature vectors of the questions. Thus, the training set comprises the vectors, each associated with a set of 29 relations. In our use-case, we apply an MLP multi-label classifier with two hidden layers including 100 and 40 units (with a ReLU[14] activation function), respectively, followed by a sigmoid activation function in the output layer, where the more details of the model are shown in Table 3.5.

3.6.3 Evaluation Metrics

Accuracy, one of the most common metrics, is used to measure the overall correctness of the employed classification models. This metric refers to the ratio of the total correct predictions over the total predictions, both correct and incorrect, as the following:

$$\text{Accuracy} = \frac{\text{correct predictions}}{\text{total predictions}} \tag{3.6.1}$$

We adopt three metrics, including recall, precision and F1-score to evaluate the effectiveness of the entire proposed approach.

Recall refers to the ratio of the correctly retrieved answers over all the given explicit standard answers; the formulation is as follows:

$$R = \frac{\text{correctly obtained answers}}{\text{gold standard answers}} \tag{3.6.2}$$

Precision refers to the ratio of the correctly retrieved answers over all returned answers, as the following:

$$P = \frac{\text{correctly obtained answers}}{\text{returned answers}} \tag{3.6.3}$$

F1-score combines precision and recall for simplicity, and the formulation is as follows:

[14]Rectified Linear Unit

$$F1 = 2 \times \frac{P \times R}{P + R} \tag{3.6.4}$$

3.6.4 Experimental Results

This section provides details on the applied models such as hyper-parameters and describes experimental results.

According to the generated questions in the offline phase, the QA dataset is made of 700 questions that are split into train and test datasets. The split ratio is 75% for the training set and 25% for the test set that is used to evaluate the employed classifiers, and then the whole proposed approach.

The XGBoost question classifier and MLP predicate classifier are trained using 5 k-fold cross-validations to control overfitting during the training. The achieved values for the accuracy of the XGBoost classifier and MLP classifier are 0.93 and 0.91, respectively.

Tables 3.4 and 3.5 show the tuned hyper-parameters of XGBoost and MLP classifiers, respectively.

As seen in Table 3.4, we opt for gbtree and auto as the booster and tree method. Since the booster is of tree type, the learning rate is kept at the default 0.3, and the number of trees is 5. Due to multi-class classification, objective should be multi:softprob. Additionally, we directly control model complexity through max_depth and gamma and add randomness to make training robust to noise by subsample and colsample_bytree. We set the max depth to 3 (3 units lower than the default 6), gamma to 20, subsample to 0.5 and colsample_bytree to 0.4.

As mentioned in Section 3.6.2, MLP classifier implementation uses two hidden layers and one output layer. The first hidden layer is made up of 100 hidden units, while the next is made up of 40 hidden units. The number of units in the input layer equals the number of the features

Table 3.4: XGBoost parameters.

Parameter	Value
booster	gbtree
learning rate	0.3
max_depth	3
number of trees	5
subsample	0.5
colsample_bytree	0.4
objective	multi:softprob
gamma	20
tree method	auto

768, and the number of units in the output layer equals the number of classes, 29. We apply the activation function ReLU for the hidden layers because ReLU is more resistant to the vanishing gradient problem than other activation functions. We use dropout in our implementation to alleviate the overfitting in MLP classifier and also apply the early stopping technique to monitor the validation accuracy. Dropout is a regularization method in the deep networks that network units are randomly ignored (dropped out) during training. Additionally, early stopping, as an effective and simple form of regularisation, stops the training once the monitored metric has stopped improving. As shown in Table 3.5, the training is carried out for 100 epochs in our MLP implementation, and the value for dropout in each hidden layer is 0.45. The learning rate (LR) controls the rate or speed at which the model learns. Once LR is large, the model learns faster but may cause undesirable divergent behaviour. Once LR is small, it allows the model to learn a more optimal but may take significantly longer to train. Here, we apply an adaptive learning rate Adam with an initial value of 0.01, considering the importance of LR in our MLP classifier. The loss function as another circuital aspect, which quantifies the difference between the expected outcome and the outcome produced by the model, is considered binary cross-entropy (it is useful for binary and multi-label classification problems).

Table 3.5: MLP parameters.

Parameter	Value
Input Dimensions	768
Epochs	100
Learning Rate	0.01
Loss Function	binary_crossentropy
Dropout	0.45
optimizer	adam
Output Dimensions	29

Table 3.6 presents recall, precision and F1-score of 175 test questions over the implemented proposed approach by the number of hops, where 'Right' denotes the number of questions that are correctly answered. According to Table 3.6, the final accuracy is 0.72, which means 72 percent of questions (127 questions out of a total of 175 questions) can be answered correctly. The F1-score has higher values for questions with fewer hops (as expected) because obtaining answers for complex questions is basically much more difficult than simple questions. Overall, the average precision, recall, and F1-score are 0.86, 0.80 and 0.82, respectively.

Table 3.6: Results on different hop questions.

Hop	Questions	Right	Recall	Precision	F1-score
1-hop	23	23	1.0	1.0	1.0
2-hop	44	34	0.86	0.84	0.84
3-hop	22	13	0.90	0.73	0.80
4-hop	86	57	0.70	0.66	0.67

We provide a failure analysis of our approach, which shows three reasons for the failure of the questions. The first reason is the question classification problem, which accounted for 6.8%. The second one is the failure of entity or class linking, which means we can not find the correct referred entity or class for 6.8% questions that are classified correctly. The third one is relation extraction, which accounted for 4.6%. Furthermore, we find that considering all the permutations to fill slots reduces the precision in some cases. In 14.8% of the questions, the queries filled with wrong permutations of entities or relations retrieve incorrect an-

swers.

Table 3.7: Failure analysis of the proposed approach.

Reason	Ratio
Question Classification	6.8%
Entity and Class Linking	6.3%
Relation Extraction	4.6%

To compare our approach with the state-of-the-art approaches, we re-implement the question classification step of TeBaQA [222]. According to the described features[15] extracted from questions in TeBaQA, we train three classifiers, including an MLP classifier, XGBoost classifier and support vector machine (SVM) classifier, to identify the query patterns of the questions. Figure 3.8 compares the gained accuracy of TeBaQA-based classifiers with the accuracy of our proposed question classifier (PQuestion Classifier). As shown, our proposed approach improves the accuracy by 12%.

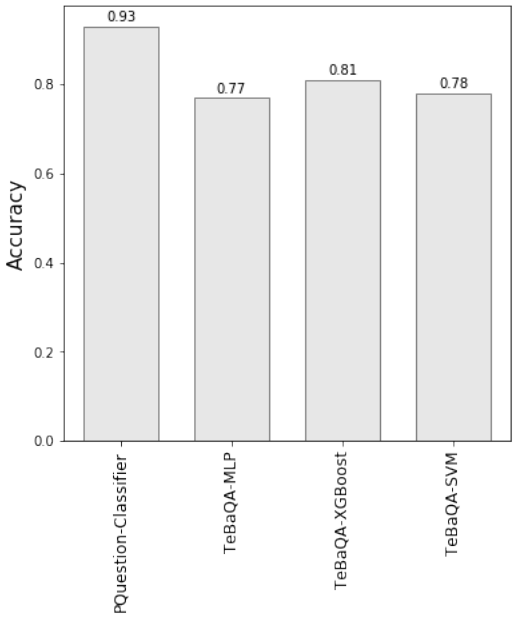


Figure 3.8: Proposed approach versus the state of the art TeBaQA result.

Note that no single KGQAS will ever be a perfect system for all the steps (question classification, entity and class linking, relation extraction

[15]The features that are not calculable in the case of Salzburgland KG are ignored.

and query execution) on all scales of KGs. For example, our proposed solution for entity linking can not be performed over a huge KG (e.g., DBPedia, which is employed in TeBaQA) with millions or billions of facts due to it will require reducing search space for each given question and then finding the relevant entities across a subset of the KG which is more likely to contain those entities. In our next research, we will develop our proposed KGQAS on the website of the underlying KG and show how end-users evaluate our system based on the answers to the questions they receive.

3.7 Discussion

In literature, most QA-based approaches either target huge KGs with large training data or define many domain-dependent rules to cover different questions. Hence, these approaches often cannot be applied directly to small and medium KGs where no training data is directly available.

From the conducted experiments, we observe that our approach achieves high performance in answering NLQs across small and medium KGs. While the approach is tested on a touristic KG, it is not restricted to a particular domain and can be generalized to other domains. Moreover, we find that answering complex questions is more challenging than simple questions, as observed in other studies. As shown in Section 3.6.4, complex questions gain less accuracy in comparison to simple questions due to different reasons, including misclassification, incorrect entity and class linking, and wrong predicted relation distributions in our work.

Our approach can be applied to SMEs' KGs, including facts ranging from thousands to hundreds of thousands. These quantities are derived from the observation made in the WordLift project and working with SMEs. For example, manufacturing SMEs can utilize our work to answer the requests of domain experts regarding the production process on top

of the semantically integrated data as a KG.

The key requirement in our work is humans' assistance to generate questions (as training data) based on their own vocabularies using the verbalized RDF triples and the gained information in the offline phase. Additionally, with complex questions, we expect to have at most four hops for reasoning over triples of a KG. This assumption is made according to the existing benchmark datasets' statistics and the practical scenarios in which it is much less likely to arrive at more than four hops. Thus, the complex questions are assumed to be 2-hop, 3-hop and 4-hop questions.

We also acknowledge the limitations of the work presented as follows:

- Since our work focuses on answering the factoid questions, it may fall short of the other types of questions (i.e., the confirmation and hypothetical questions, discussed in Section 3.2).

- While our approach takes into account all the possible RDF templates, it excludes questions that are rarely posed because of their low frequency. Hence, we take out query patterns with a few examples when categorizing the generated questions in the offline phase. This exclusion leads to limited misclassifications in the question classification step.

- In the slot filling and query execution step, incorrect permutations of detected entities, classes and relations can increase false positives and consequently decrease precision.

With the above highlighted limitations, the task of QA over small and medium scaled KGs can still be significantly improved.

3.8 Conclusions

This paper proposes an isomorphism-based approach to develop KGQASs over small and medium scaled KGs, including the offline and semantic parsing phases. The offline phase targets at generating NLQs and their answers over the underlying KG using RDF templates in a semi-automated way. The outcome of this phase is utilized as training data in the semantic parsing phase. According to the grammatical structure of NLQs, an XGBoost classifier is trained to predict the relevant query pattern of a given input question in the semantic parsing phase. The query patterns comprise entity, class and relation holders as slots. We define a similarity score based on the string and semantic measurements to select the appropriate entities and classes from the KG. Additionally, a multi-label MLP classifier is applied to predict probability distributions for relations and then extract the relations in the question sentence. Then, the Cartesian product of the possible values in each slot is considered to derive and return answers.

To show how the proposed approach is applied in practical use-cases, we use SalzburgerLand KG describing touristic entities and achieve the average recall, precision and F1-score of 0.86, 0.80 and 0.82, respectively.

Looking back at the main research sub-questions and challenges, we address the training data problem in small and medium scaled KGs through the offline phase. We determine query patterns using graph isomorphisms from SPARQL queries to deal with the task of defining hand-crafted templates. Different similarity measurements and probability distributions tackle the linguistic gap between question sentences and KGs' vocabularies.

The future work will focus on the multilingual aspect and show how the proposed approach can be adapted to a KG with a new language (in particular, the German version of SalzburgerLand KG is also avail-

able). To evaluate the proposed KGQAS using a practical setting, we will develop the system on the website and ask the end-users to assess the system. Furthermore, to generate questions in the offline phase, we will involve more people to increase the diversity of the generated questions. Also, "yes" or "no" questions and factoid questions merged with complex aggregation functions (e.g., "How many ski-resorts with more than 100 slopes operate between October 20 and 31?") will be covered in the next investigation.

Chapter 4

Building Knowledge Subgraphs in Question Answering over Knowledge Graphs

Contents

4.1 Abstract

Question answering over knowledge graphs targets to leverage facts in knowledge graphs to answer natural language questions. The presence of large number of facts, particularly in huge and well-known knowledge graphs such as DBpedia, makes it difficult to access the knowledge graph for each given question. This paper describes a generic solution based on Personal Page Rank for extracting a small subset from the knowledge graph as a knowledge subgraph which is likely to capture the answer of the question. Given a natural language question, relevant facts are determined by a bi-directed propagation process based on Personal Page Rank. Experiments are conducted over FreeBase, DBPedia, and WikiMovie to demonstrate the effectiveness of the approach in terms of recall and size of the extracted knowledge subgraphs.

Keywords: Knowledge graphs, Question answering systems, Knowledge subgraph, Personal Page Rank

Publication: Sareh Aghaei, Kevin Angele, and Anna Fensel. "Building knowledge subgraphs in question answering over knowledge graphs". In: Proceedings of the 22nd International Conference on Web Engineering. Bari, Italy: Springer-Verlag, 2022, pp. 237–251. DOI: 10.1007/978-3-031-09917-5_16.

4.2 Introduction

With the growth of the data web, a massive amount of structured data has become available on the web in the form of knowledge graphs (KGs). To assist end users to access KGs, knowledge graph-based question answering systems (KGQASs) have emerged to answer natural language questions [208, 235, 236]. Although large KGs such as DBPedia with millions or billions of facts are ideal sources for answering questions, accessing these KGs for each given question has become an intricate chal-

lenge. To overcome this challenge, the recent KGQASs extract a subset from the KG namely a knowledge subgraph for the question posed over the KG as illustrated in Figure 4.1.

A knowledge subgraph targets to prune irrelevant parts of the KG's search space and contains only a set of facts that is likely to capture the answer of a given question. Reducing the search space plays a key role in the efficiency of different types of KGQASs including (1) rule-based, (2) information retrieval-based, and (3) semantic parsing-based systems (discussed in Section 4.3). Knowledge subgraphs lead to reducing manual works required for setting up the rule-based systems [6, 211, 212], pruning candidate entities and reducing training cost in the retrieval-based systems [7, 8, 237] and making improvements in the mapping stages of semantic-parsing systems due to preventing unnecessary computations [238].

Thus, the task of building knowledge subgraphs over huge KGs avoids exploring the whole KG for each question in KGQASs and narrows down the search space. Basically, a trade-off between answer presence and search space size [23] is required to build knowledge subgraphs. For example, the mean shortest path between entities in DBpedia is around 5-hops, so extracting relevant subgraphs only by navigating a predefined number of hops from a set of entities that represent the question's focus leads to a big part of the DBpedia however covers the answers, as an instance, given a simple question such as "Where is the capital of the US?", there is approximately 600K facts around 1-hop of the US's entity in DBPedia. In contrast, to further reduce the retrieved facts, commonly used techniques [7, 9, 239] even fail to capture answers of some simple questions that can be addressed through one fact (discussed later).

Therefore, the primary research question of this paper is *how to extract a knowledge subgraph for a posed natural language question that*

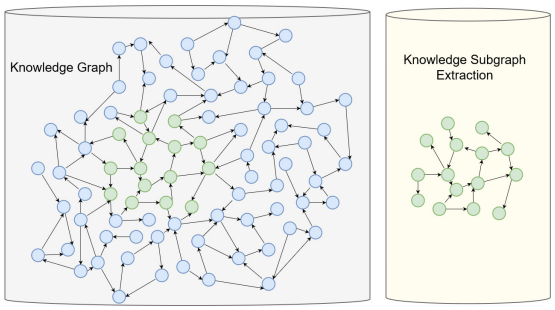


Figure 4.1: Extraction a subset from the knowledge graph.

reduces the size of the KG significantly and covers the answer. For example, given the question sentence "Give me all the companies with more than 1000 employees that were founded in the US from 1986 to 2000" over DBPedia, the extracted knowledge subgraph has to contain relevant facts around the entity of "the US" from millions facts stored in DBPedia which cover the foundation date and employee number of the companies located in the US. Note that the state-of-the-art KGQASs require to learn models for mapping the question to DBPedia facts to find the answer, where the extracted knowledge subgraph helps these systems to tackle with the huge search space size of DBPedia.

A general architecture to construct a knowledge subgraph for each question to avoid exploring the whole KG is shown in Figure 4.2. The architecture consists of three main steps namely topic entity identification, neighborhood retrieval, and knowledge subgraph retrieval. The topic entity identification step employs entity linking (EL) to recognize named entities of questions which reflect the major focus of the questions and next map each entity mentioned in the questions to its corresponding entity in the KG (known as topic entity). Then, the neighbors around topic entities need to be retrieved through n-hop reasoning over the underlying KG. Finally, a knowledge subgraph which includes the topic entity as its first entity, is expanded based on various techniques such as heuristics, neural networks across the retrieved neighbors.

Personal Page Rank (PPR) [25] as a heuristic query-dependent tech-

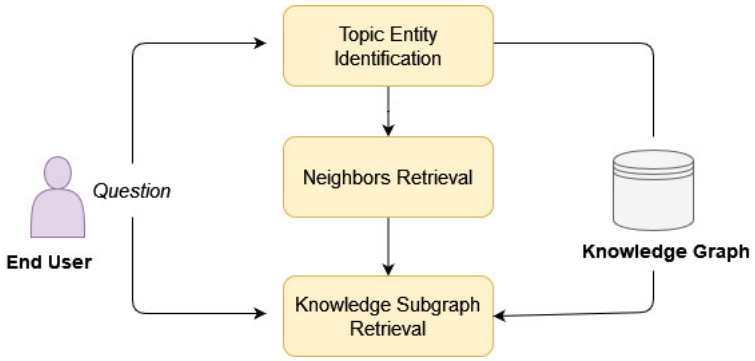


Figure 4.2: General architecture of knowledge subgraph construction.

nique is widely used in KGQASs to build a knowledge subgraph around the topic entity with respect to the natural language question posed by the end user [7, 9, 210, 239]. This paper follows the research of [7] in using PPR and proposes a bi-directed propagation technique, called BiDPPR to compute relevance scores for nodes. The BiDPPR employs a bi-directed iterative process in which the scores are propagated through incoming and outgoing edges of nodes in each iteration. The major novelty of the proposed approach lies in detecting when there is no directed path from topic entities to answer entities, the PPR technique fails to build subgraphs covering the answer entities and then proposing a solution to deal with it. For example, given posed questions "Where does Piccadilly start?" and "Where was the author of the theory of relativity educated?" over WikiData and DBPedia, respectively, PPR technique fails to retrieve the knowledge subgraphs which cover the answers because there are no direct paths from topic entities ("Piccadilly" and "theory of relativity") to answer entities ("Dover street" and "ETH Zurich") over the underlying KGs as shown in Figure 4.3. Note that although the question "Where does Piccadilly start?" only needs one fact to be answered, the PPR-generated knowledge subgraph does not include the answer.

The main contributions of the paper can be summarized as follows:

1. An approach to build knowledge subgraphs over KGs for questions is proposed which follows the generic existing architecture shown

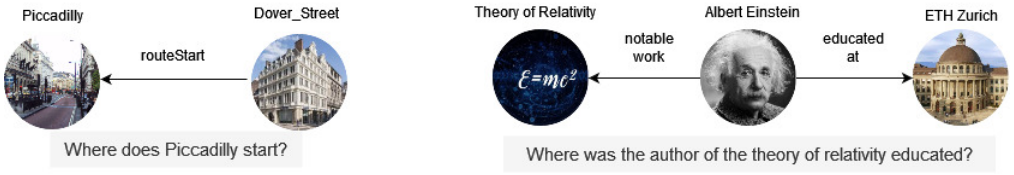


Figure 4.3: The path between topic entity and answer entity.

in Figure 4.2.

2. A new bi-directed propagation technique based on PPR is introduced to retrieve those entities from a KG which are more likely to answer questions.

3. Experimental results are demonstrated on QA datasets over FreeBase, DBPedia and WikiMovie and a comparison with available solutions to prove the effectiveness of the proposed approach in terms of recall. Furthermore, the results show how the proposed solution contributes to extracting smaller knowledge subgraphs.

The remind of the paper is organized as follows. Section 4.3 provides an overview on the related works. The proposed approach is discussed in Section 4.4, and Section 4.5 provides a detailed experimental evaluation including a comparison against state-of-the-art solutions. Finally, Section 4.6 concludes the paper and gives directions for future research.

4.3 Related Work

The research progress on building knowledge subgraphs in question answering (QA) over KGs can be divided into three categories including filtering-based techniques, heuristic-based techniques, and neural-based techniques.

1. **Filtering-based techniques** rely on predefined rules to filter the number of facts around topic entities. The definition of rules leads

to limited scalability and researchers and developers require familiarity with the underlying scheme's KG. Moreover, these techniques are not able to significantly prune irrelevant entities. The introduced Graph Alignment Question Answering (GAQA) approach in [238], defines some query patterns and leverages users' interceptions through an interface to determine the number of required hops to retrieve the paths in the KG. Then, each given question is mapped into a query pattern according to the identified required hops. To prune unnecessary facts while avoid knowledge loss for answering the question, three filtering functions are inserted into the query patterns: (1) filtering out unnecessary predicates (e.g., predicates `http://dbpedia.org/ontology/wikiPageID`, `http://dbpedia.org/ontology/abstract` are assumed as unnecessary predicates in DBPedia KG), (2) filtering out unnecessary literal leaf nodes (e.g., the nodes with irrelevant language tags have to be eliminated), and (3) filtering out unnecessary resource nodes (e.g., a set of unnecessary namespace URI is defined and resource nodes which belong to this set, are filtered). Finally, a SPARQL query is executed according to the mapped query pattern and the returned result is considered as the knowledge subgraph.

2. **Heuristic-based techniques** use heuristics to build a knowledge subgraph. The PPR [25] as a heuristic algorithm is widely applied in recent KGQASs to retrieve relevant facts around questions [239, 210, 9, 7].

   The PageRank-Nibble (PRN) [240] is an approximate of PPR by applying a tolerance threshold (ϵ) which is used in [7]. Firstly, the topic entity is assumed as query node and all the paths with a maximum length starting at the topic entities are retrieved as a neighborhood graph. Then, the adjacency matrix of the neigh-

borhood graph as a directed graph is generated based on the edge weights. The edge weight is calculated based on the similarity between the edge's surface form[1] and the question. To find the similarity between the question and the edge, GloVe[2] is applied to obtain vector representations and the cosine similarity between two vectors is calculated. Then, the initial PRN score of the topic entity is set to 1 and the other nodes are set to 0. Next, through an iterative process, the PRN score of nodes are computed. In each iteration t, the PRN score is propagated through the outgoing edges of the nodes. After T iterations, the k-top nodes with highest PRN scores (their scores are greater than ϵ) with edges among them are selected as the more relevant facts to the question. The main issue is that PRN fails to retrieve the answer entities once there is no directed path from topic entities to answer entities. The introduced approach in [9] follows the same idea in [7] and expands one hop for CVT[3] (Compound Value Type) entities in Freebase to obtain the extracted knowledge subgraphs (this expansion is applicable if the KG includes CVT nodes).

3. **Neural-based techniques** utilize neural networks to build a subgraph that contains facts relevant to a given question. The Pullnet [8] fulfills an iterative process to construct a subgraph. In each iteration, a graph convolutional network (graph CNN) is used to identify nodes that should be expanded using the pull operation. The pull operation retrieves the top facts from the KG around entity e which are constrained to have e as their subject or object. The retrieved facts are ranked based on the similarity between the fact's relation and the question using a classifier. Thus, the clas-

[1]The surface form of an edge is the value of rdfs:label if the edge does not have a label, the variable part of its URI is adopted as the surface form.

[2]`https://nlp.stanford.edu/projects/glove/`

[3]`https://developers.google.com/freebase/guide/basic_concepts#cvts`

sifier predicts which retrieved facts are more relevant to the question. The major challenge of these techniques is the requirement for question-answer pairs as training data.

Current KGQASs can be classified as (1) rule-based, (2) information retrieval-based and (3) semantic parsing-based systems. In rule-based systems, much manual work is required in the preparation phase due to mappings from recognized entities to predefined queries or rules. Then, those queries or rules are evaluated over the underlying KG to retrieve the expected answer [211, 6, 212]. Extracting knowledge subgraphs reduces the manual work required for setting up a rule-based KGQAS. The information retrieval-based systems need to retrieve all candidate answers and then rank them to select the most pertinent answer. So, building a small knowledge subgraph can help pruning the candidate entities and improving the performance of the system [7, 8, 237]. KGQASs based on semantic parsing basically convert questions to executable queries. In these systems, the unstructured question is mapped to intermediate logical forms and then the intermediate forms are transformed into queries, such as SPARQL. Obviously, reducing the search space on KGs through constructing a pruned knowledge subgraph based on the input question makes improvements in mapping stages of semantic-parsing KGQASs [238].

Although the stream of research on QA over KGs has gained the solutions for building knowledge subgraphs, the recall and size of knowledge subgraphs still need to be improved. For example, filtering-based techniques are not effective in reducing size from a large KG such as DBPedia, PRN fails in building high-recall knowledge subgraphs once there are no directed paths from topic entities to answer entities, and neural-based techniques demand training question-answer pairs which are not available in many practical settings. This paper proposes a bi-directed propaga-

tion technique based on PPR (BiDPPR) to build high-recall knowledge subgraphs by considering incoming edges of nodes as well as outgoing edges while the size of the extracted subgraphs not being larger than those constructed by PRN.

4.4 The Approach

This section presents the proposed approach for constructing high-recall knowledge subgraphs according to the generic architecture shown in Figure 4.2 that comprises three main stages including topic entity identification, neighborhood retrievals and knowledge subgraph retrieval.

4.4.1 Topic Entity Identification

The task of EL is to link an entity mentioned in a text corpus to the corresponding entity in a knowledge base [241]. Here, given a KG containing a set of entities and a set of questions, the goal of EL is to map each entity mentioned in questions to its corresponding entity in the KG [242, 58]. The corresponding entities (known as topic entities) generally show the topic of the given question sentences. In this paper, the topic entities of questions are identified through existing EL tools including DBpedia Spotlight and S-MART. The DBpedia Spotlight system [223] automatically annotates questions' sentences with DBpedia URIs, and S-MART is applied for entity linking in FreeBase. This paper assumes that there is at least an entity mentioned in each question (known as topic mention), which shows the main focus of the question and EL identifies its mapping entity in the KG. As an example, given the question "Give me all the companies with more than 1000 employees that were founded in the US from 1986 to 2000", the named entity "the US" is the topic mention which is mapped to `http://dbpedia.org/resource/United_States` as

the topic entity.

4.4.2 Neighborhood Retrieval

Once the topic entity of the question is identified, all the entities in the underlying KG which have a distance (distance between two nodes is the number of edges in a shortest undirected path) smaller or equal n are extracted. The extracted entities along with relations among them are defined as neighborhood graph which consists of the n-hop neighbors around the topic entity (according to Definition 1). Generally, according to the number of required hops for reasoning over facts, questions can be grouped into two categories: simple questions and complex questions. A simple question, namely single-hop question, can be answered through only one fact whereas a complex question, called multi-hop question, requires reasoning over two or more facts of the KG [210, 10]. Since, in real scenarios, the maximum length of path starting at topic entity do not exceed 3 in general [238], this paper considers $n = 3$.

Definition 1 A neighborhood graph is defined as $G_N = (N_N, E_N)$ where N_N is a set of entities around the topic entity T_e with distance $d <= n$ from T_e (distance between two nodes is the number of edges in a shortest undirected path), E_N is a set of edges with distance $d < n$ from T_e and n is the depth (the longest undirected path from the topic entity) of the graph.

To build neighborhood graphs with maximum depth n, SPARQL[4] patterns are defined according to n and T_e. Basically, the total number of possibilities to construct SPARQL patterns around the topic entity T_e with depth n is 2^n. Therefore, 2, 4 and 8 SPARQL patterns can be defined for depths 1, 2 and 3, respectively (the topic entities are shown in blue colour). Figure 4.4 illustrates all the possible states to construct

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

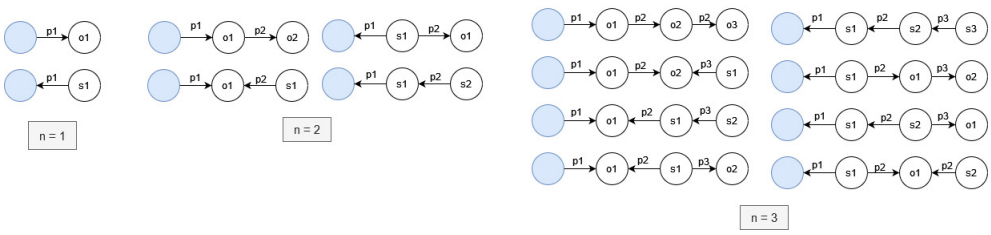


Figure 4.4: Possible states to construct SPARQL patterns.

```
SELECT <T_e> ?p1 ?o1. ?o1 ?p2 ?o2
        WHERE {
      <T_e> ?p1 ?o1 .
      ?o1 ?p2 ?o2
           }

SELECT <T_e> ?p1 ?o1. ?s1 ?p2 ?o1
        WHERE {
      <T_e> ?p1 ?o1.
      ?s1 ?p2 ?o1
           }

SELECT ?s1 ?p1 <T_e>. ?s1 ?p2 ?o1
        WHERE {
    ?s1 ?p1 <T_e>.
      ?s1 ?p2 ?o1
           }

SELECT ?s1 ?p1 <T_e>. ?s2 ?p2 ?s1
        WHERE {
    ?s1 ?p1 <T_e>.
      ?s2 ?p2 ?s1
           }
```

Figure 4.5: SPARQL patterns with depth 2.

SPARQL patterns with depth $n <= 3$ and Figure 4.5 shows the SPARQL patterns when n is equal to 2.

4.4.3 Knowledge Subgraph Retrieval

After creating a neighborhood graph for a given input question, a knowledge subgraph is retrieved around the topic entity across the neighborhood graph according to BiDPPR. The formal definition of a knowledge subgraph is provided in Definition 2.

Definition 2 A knowledge subgraph is a subset of the neighborhood graph which can be defined as $G_K = (N_K, E_K)$ where $N_K \subset N_N$ and $E_K \subset E_N$ and N_K includes the entities which are more likely to be answer entities.

The proposed technique, BiDPPR, tackles the issue of lacking a directed path from topic entity to answer entity in PRN through a bi-directed propagation process which is summarized as following:

- To consider the impact of incoming edges of a node during the

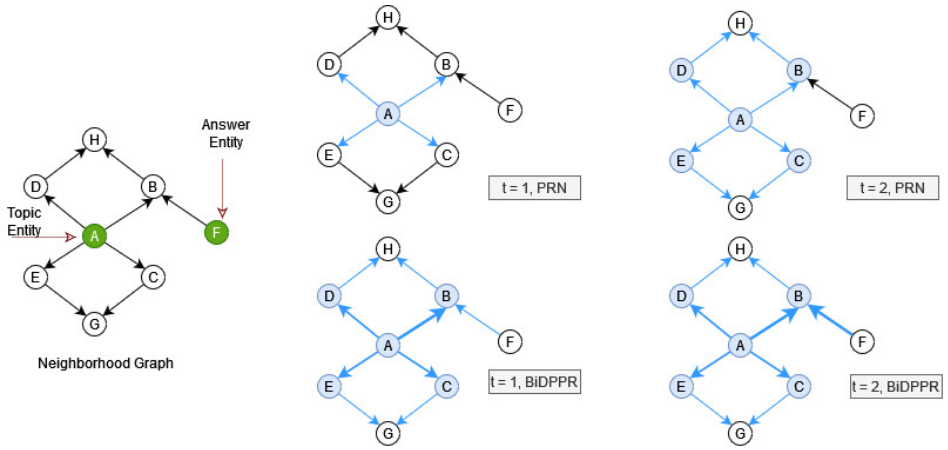


Figure 4.6: Propagation process from the topic entity to the answer entity using PRN and BiDPPR.

propagation process as well as its outgoing edges, a linear combination of propagation along outgoing edges and incoming edges is utilized to find the BiDPPR score of nodes. If M denotes the adjacency matrix of the neighborhood G_N which presents the edge weights then the transpose of M can be considered as a matrix that includes the inverse relations between entities and let this matrix be M^T. Thus, the calculation of BiDPPR is formulated as:

$$pr_v^{(t)} = (1-\alpha)pr_v^{(t-1)} + \alpha(\omega_1 \sum_{<n,r,v>} w_r.pr_n^{(t-1)} + \omega_2 \sum_{<v,r,n>} w_r^{(t)}.pr_n^{(t-1)}) \quad (4.4.1)$$

Where w_r and w_r^t denote the weights of the edge r in both directions based on the adjacency matrix M and transpose of adjacency matrix M^T, respectively. Also, ω_1 and ω_2 are assumed as coefficient ratio for the incoming edges and outgoing edges, respectively.

- To compute the adjacency matrix, similar to [7], pretrained word embeddings GolVe is applied to generate the embedding of the question and the edges' surface forms. The cosine similarity between the embeddings of the given question and the edge is considered as

the weight of that edge.

- To preserve the origin direction of edges, the impact of propagation along outgoing edges ω_1 should be greater than the impact of propagation along incoming edges ω_2.

- In the first initialization, BiDPPR scores are set to $\frac{1}{|N_N|}$ for all non-topic entities and the BiDPPR scores of topic entities are set to $1+\frac{1}{|N_N|}$ (Equation 4.4.2). Furthermore, the scores are normalized after each iteration to prevent any explosion.

$$pr_v^{(0)} = \begin{cases} \frac{1}{|N_N|} + 1 & topic\,entities \\ \frac{1}{|N_N|} & otherwise \end{cases} \tag{4.4.2}$$

Similar to PRN, the k-top nodes by BiDPPR score, along with edges among them are selected to make the knowledge subgraph after T iterations. It is noticeable that the sizes of extracted knowledge subgraphs do not increase in comparison to the extracted subgraphs by PRN. The size of the knowledge subgraph is dependant on K as well as ϵ. In Section 4.5 the coverage of PRN and BiDPPR for different values of K are compared.

Figure 4.6 illustrates the propagation process in PRN and BiDPPR in a sample neighborhood graph without any directed path from the topic entity A to the answer entity F. As shown in Figure 4.6, in the first iteration $t = 1$, the PRN score will be 0 for all nodes except the topic entity A, and the propagation will only happen from node A (the edge are shown in blue colour). For $t = 2$, the PRN score will be non-zero for node A and its neighbors including B, C, D and E, and the propagation will happen from these nodes. For next iterations, the PRN score will be non-zero for the nodes A, B, C, D, E, G and H. Since H and G as dead nodes have no outgoing edges, their scores can not be propagated in the

graph. Thus, the PRN score for the node F will stay at 0 by the end due to lack of a directed path from node A to node F. While in BiDPPR, the propagation does not start from a specific node (as the initial scores are not zero) however node A as the topic entity (with the initial score $\frac{1}{8} + 1$ according to 4.4.2) significantly impacts on its neighbors. Since the propagation spreads out in both directions in BiDPPR, the score of node F will increase remarkably in the next iteration ($t = 2$) due to happening propagation along incoming edge of node B (note that the weight of the edge between F and B has to be high because its weight is computed based on cosine similarity between the embedding of the question sentence and the edge's label).

4.5 Experiments

In this section, the proposed approach is evaluated on Freebase, DBpedia and WikiMovies [243] with three QA-benchmarks separately. The code[5] is implemented in python and Stardog[6] is utilized to set up SPARQL endpoints. The PRN technique with $\epsilon = 1e - 6$ is performed.

4.5.1 Knowledge Graphs

Freebase is a practical, scalable KG used to structure general human knowledge [141]. It was launched by Metaweb as an open, public and collaborative KG with schema templates for most kinds of possible entities such as persons, cities, movies, etc in 2007.

DBPedia is extracted from structured data in Wikipedia through a crowd sourcing community that the main idea behind the extraction is using the key-value pairs in the Wikipedia infoboxes.

[5]The GrafNet repository on the Github is reused according to the proposed approach.

[6]`https://www.stardog.com/get-started/`

WikiMovies is the QA part of the Movie Dialog dataset and supports three different settings of knowledge including (1) using a traditional knowledge base (KB), (2) using Wikipedia as the source of knowledge, and (3) using information extraction over Wikipedia.

4.5.2 QA Datasets

WebQuestionsSP(WebQSP) dataset [230] includes 4737 natural language questions that were produced by crawling the Google suggest API [228] and are answered through Freebase entities. The questions need up to 2-hop reasoning from the KG. Moreover, the questions are more colloquial and biased towards topics that are frequently asked from Google [215, 7].

QALD-6 [244] is the sixth installment of the QALD (Question Answering over Linked Data challenge) and focuses on questions which need up to 3-hop reasoning from the DBPedia. QALD-6 includes 350 training questions and 100 test questions which the test dataset is applied in this experiment.

MetaQA dataset [243] is a large-scale multi-hop dataset in the domain of movies. It includes more than 400k 1-hop, 2-hop and 3-hop questions, containing three individual datasets namely, MetaQA-1hop, MetaQA-2hop and MetaQA-3hop [237].

4.5.3 Metric

The number of entities in knowledge subgraphs is considered as a metric to compare sizes of knowledge subgraphs. Furthermore, recall as a classical metric to evaluate the effectiveness is adopted for showing the coverage of the constructed knowledge subgraphs. Here, recall is the fraction of the answers that are successfully retrieved by the subgraph as the following:

Table 4.1: Results on WebQSP, QLAD-6 and MetaQA with 500 entities.

Dataset	PRN	BiDPPR
WebQSP	89.9	92.2
QLAD-6	62.7	84.8
MetaQA-1hop	100	100
MetaQA-2hop	100	100
MetaQA-3hop	83.0	92.2

$$recall = \frac{retrieved\ entities \cap answer\ entities}{answer\ entities} \tag{4.5.1}$$

4.5.4 Results

The experimental results for WebQSP, QLAD-6 and MetaQA datasets with 500 entities ($k = 500$) are shown in Table 4.1. On WebQSP dataset, the number for recall in PRN is 89.9%, this increased to 92.2% in BiDPPR. The BiDPPR is comparable to the PRN on QLAD-6, the recall improves by 22.1%. On MetaQA dataset, BiDPPR shows the recall improvement around 10% over 3-hop questions. In the case of MetaQA-1hop and MetaQA-2hop, both techniques achieve fully-coverage knowledge subgraphs.

To illustrate that BiDPPR obtains higher recall knowledge subgraphs with fewer number of entities in comparison to PRN, WebQSP is selected as (1) WebQSP includes much more questions in compassion to QALD-6, and (2) Freebase is far larger than WikiMovies. Figure 4.7 presents the recalls of PRN and BiDPPR on WebQSP over the size of knowledge subgraphs. As the graph shows, the coverage of BiDPPR retrieval knowledge subgraphs is relatively quickly in comparison to PRN. For example, the number of entities to archive the recall 92.0% in PRN is $k = 1200$ while BiDPPR is able to achieve the same recall with $k = 500$.

One point to note is that BiDPPR uses the transpose of the adjacency matrices to consider the inverse direction of the relations. Since the transpose of a matrix can be done in $O(1)$ time (and space), BiDPPR

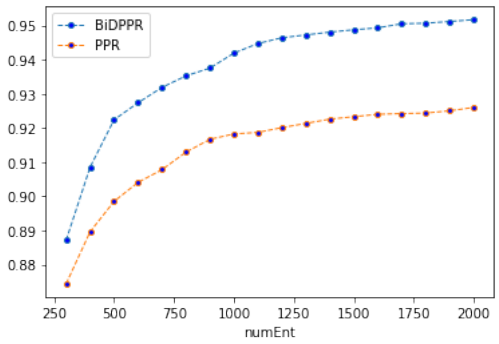


Figure 4.7: Recall of BiDPPR and PRN on WebQSP with different number of entities.

Table 4.2: Results on WebQSP with 2000 entities.

Technique	PRN	PRN+CVT	BiDPPR
Coverage	92.6	94.9	95.2

does not affect the time complexity of PRN[7].

4.5.5 Compared Approaches

BiDPPR is compared with PRN (used in [7, 9]) according to Table 4.1. It is seen that BiDPPR finds higher coverage and smaller knowledge subgraphs for questions.

According to [9], the recall of PRN on WebQSP can increase to 94.9 with 2000 entities once the extracted subgraphs are expanded one hop for CVT entities in Freebase however BiDPPR gains the recall 95.2 with the same number of entities as shown in Table 4.2.

The results reported by GAQA in [238] give the answer recall when answer entities are retrieved over the extracted knowledge subgraphs[8] and the coverage of the constructed knowledge subgraphs is not shown in its paper. Basically, GAQA can achieve full-coverage knowledge subgraphs if the query patterns are correctly identified due to it only filters the obvious unnecessary items (e.g, the predicates which are mainly used

[7]Time complexity of PRN is O(m*n) [m = no. of iterations, n= no. of nodes]

[8]After constructing knowledge subgraphs, GAQA obtains answers of given questions over the extracted subgraphs based a graph-alignment method and then reports the results.

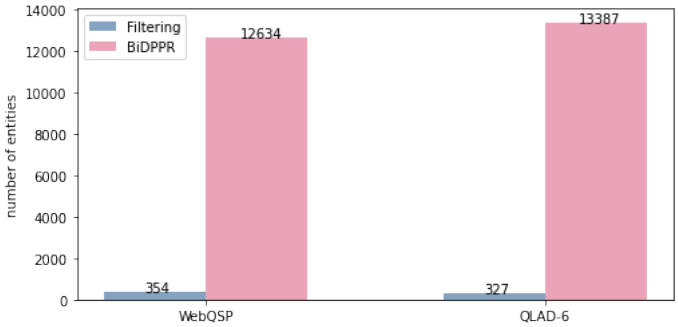


Figure 4.8: Comparison of the knowledge subgraphs's size in BiDPPR and GAQA.

to link the KGs) however the knowledge graphs are significantly larger than those generated by BiDPPR. Since GAQA's source code is not publicly available, this research study re-implements GAQA's solution. In this re-implementation, 15 questions are randomly selected from each dataset (WebQSP and QLAD-6) and their query patterns are identified based on their SPARQL queries [9]. Figure 4.8 depicts the average size of the knowledge subgraphs (in terms of the number of entities) for the randomly selected questions and it is clearly shown that BiDPPR builds substantially smaller knowledge subgraphs. Furthermore, the recalls of the retrieved subgraphs by BiDPPR are 95.0% and 0.89 for the selected questions in WebSQP and QALD-6, respectively.

Compared to PullNet, BiDPPR needs no training data. PullNet has to train a classifier based on question-answer pairs to predict the relevant facts to questions. The results of PullNet are not directly comparable to the results of this paper due to PullNet's results show the recall after obtaining answers over the knowledge subgraphs and the source code is not available as well. However, according to [8], PullNet is able to retrieve far fewer entities with higher recall in comparison to PRN.

[9]The task of identifying query pattern needs end users' assistance in GAQA

4.6 Conclusion

With the increasing growth of KGs, QA over KGs can be seen as the most promising approach to make the KGs easily accessible for end users. Since a KG is typically large and stores millions of facts, accessing the KG for each given question in KGQASs is difficult or even impossible. Extracting a small subset from the KG (known as knowledge subgraph) that is likely to contain the answer entity, defiantly reduce the search space and make the final answer extraction process easier. This paper proposes an approach including three major stages: topic entity identification, neighborhood retrieval and knowledge subgraph retrieval. The main focus of the approach is to introduce a new derivation of the PPR technique called BiDPPR to construct the knowledge subgraphs. Once there is no directed path from topic entities to answer entities, PPR technique fails to construct knowledge subgraphs which contain the answer entities. To address this problem, BiDPPR suggests propagating along the incoming edges as well as the outgoing edges. The proposed approach finds higher recall knowledge subgraphs with fewer entities than the ones created before. The effectiveness of the proposed approach in terms of recall and size is illustrated on WebQuestionsSP, QLAD-6 and MetaQA datasets which apply Freebase, DBPedia and WikiMovie as KGs to answer questions, respectively.

In the future, given a natural language question, a syntactic-semantic representation is created as question graph and the number of hops to retrieve the neighborhood graph is calculated based on the longest path in the question graph. Then, the task of QA over KGs is reduced to finding subgraph matches of the question graph over the knowledge subgraph.

Chapter 5

Question Answering over Knowledge Graphs: A Graph-Driven Approach

Contents

5.1 Abstract

With the growth of knowledge graphs (KGs), question-answering systems make the KGs easily accessible for end-users. Question answering over KGs aims to provide crisp answers to natural language questions across facts stored in the KGs. This paper proposes a graph-driven approach to answer questions over a KG through four steps, including (1) knowledge subgraph construction, (2) question graph construction, (3) graph matching, and (4) query execution. Given an input question, a knowledge subgraph, which is likely to include the answer is extracted to reduce the KG's search space. A graph, named question graph, is built to represent the question's intention. Then, the question graph is matched over the knowledge subgraph to find a query graph corresponding to a SPARQL query. Finally, the corresponding SPARQL is executed to return the answers to the question. The performance of the proposed approach is empirically evaluated using the 6th Question Answering over Linked Data Challenge (QALD-6). Experimental results show that the proposed approach improves the performance compared to the-state-of-art in terms of recall, precision, and F1-score.

Keywords: Knowledge Graphs, Question Answering, Graph Matching, SPARQL

Publication: Sareh Aghaei, Sepide Masoudi, Tek Raj Chhetri, and Anna Fensel. "Question Answering over Knowledge Graphs: A Graph- Driven Approach". In: Proceedings of the 21st IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology. forthcoming. Niagara Falls, Canada, 2022.

5.2 Introduction

With the steady growth of web data, knowledge graphs (KGs) have been increasingly adopted to structure and store large-scale data on the web.

A KG, presented in the Resource Description Framework (RDF)[198], often includes millions or billions of facts in the form of subject-predicate-object triples.

While SPARQL[1] is a standard way to access RDF KGs, end-users find it difficult due to the complexity of the SPARQL syntax and the RDF schema [208, 245, 246]. To tackle this drawback, question answering (QA) over KGs has emerged. A KG-based question answering system (KGQAS) targets to automatically answer natural language questions (NLQs) posed by end-users.

Parsing NLQs and obtaining answers present numerous challenges. Some of the key challenges are: (1) large search space in KGs; (2) mapping unstructured NLQs to structured queries without losing the intention of the NLQs; and (3) the linguistic gap between questions and KGs' vocabularies.

To address the mentioned challenges and improve KGQASs' performance, this paper proposes a graph-driven approach focusing on factoid[2] questions, both simple and complex.

Based on our objective, which is to improve the performance of QA over KG, our work seeks to answer the following key research questions: (1) how to extract a knowledge subgraph for a posed NLQ that reduces the size of the KG while covering the answer?; (2) how to bridge the gap between NLQs and SPARQL queries in a semantic parsing based KGQAS?; and (3) how to take advantage of the task of graph matching to match question graphs and knowledge subgraphs?.

Inline with our research questions, this paper makes the following contributions:

- We extend our previous research study to reduce the search space

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

[2] The factoid questions are factual in nature and commonly start with a WH-word such as when, where, who, what, etc.

in KGs that introduces a new derivation of the personal page rank (PPR) algorithm, named Bi-Directed PPR (BiDPPR) [24].

- The syntactic structure of the question and a set of effective transformation rules are used to present question's intentions in graph structures.

- The task of graph matching between question graphs and knowledge subgraphs is reduced to node matching and edge matching.

- Extensive experiments are conducted to validate the performance of the proposed approach in terms of recall, precision and F1.

In our study, we conceptualize the task of QA over KGs to parse NLQs and then convert the questions into SPARQL queries. Even though QA over KGs remains a challenging task, our approach outperforms state-of-the-art technologies, as demonstrated by our findings (see Section 5.5.3).

The remainder of the paper is organized as follows. Section 5.3 provides an overview of the related works. The proposed approach is described in Section 5.4, and Section 5.5 reports a detailed experimental evaluation, including a comparison against state-of-the-art solutions. Finally, Section 5.6 concludes the paper and gives directions for future research.

5.3 Related Works

In this section, we provide a review of the related works. First, we provide an overview of the different types of QA over KGs (see Section 5.3.1). Thereafter, we discuss the related works, focusing on our study, which is graph-based QA approaches (see Section 5.3.2).

5.3.1 Types of QA over KGs

1. *Template-based methods:* Template-based methods rely on predefined templates (a pseudo-query with some slots) to parse questions and provide logical forms [4, 5, 6, 213, 247]. The significant drawbacks of template-based systems are limited scalability and low flexibility of templates.

2. *Information retrieval-based methods:* These methods extract a subgraph with respect to a NLQ and treat the subgraph's nodes as candidate answers. Then, the semantic relevance of the question and candidate answers are computed according to the features extracted from the question and encoded candidate answers. Finally, the candidate with the highest matching score is selected as the final answer [248]. While these solutions can handle complex reasoning on a graph structure and fit into end-to-end training, the intermediate reasoning is less interpretable, and a considerable amount of training data is required [7, 8, 9].

3. *Semantic parsing-based methods:* This category aims to build a semantic parser that transforms an input NLQ into an intermediate form and then converts the intermediate form into a logical form. There are two mainstream branches to generate intermediate forms: (a) encoder-decoder-based techniques and (b) graph-based techniques [10]. The encoder-decoder techniques [12, 145, 249] utilise trees or high-level programming languages to represent questions' intentions which require a lot of annotated training examples [11]. The graph-based methods use graphs to bridge the gap between the questions and logical executable forms.

5.3.2 Graph-based QA Approaches

The introduced system in [207], known as the graph answer (GAnswer), obtains two frameworks, namely node-first framework (NFF) and relation-first framework (RFF). In NFF, nodes of a given question are found and then the edges to connect nodes are identified to construct a graph. This framework finds candidates for each node and edge of the graph using two dictionaries. Once the candidate lists are provided, a button-up process is used to find approximate matches over the KG. Different from NFF, RFF begins with extracting semantic relations from a NLQ and then looks for the associated nodes based on heuristic rules. A user-guided graph similarity approach is presented in [238], named the graph alignment question answering (GAQA) framework. GAQA receives a question graph constructed by the end-user and builds a knowledge subgraph by filtering out unnecessary facts. An alignment-based method formulates the pairwise graph alignment with integer linear programming. The outcome of this alignment is converted to a structured query to retrieve the answer.

Since our approach does not rely on predefined templates, it is more robust in answering different questions than template-based techniques. Different from information retrieval-based techniques, our approach does not demand annotated training data and produces a more interpretable reasoning process by generating intermediate logic forms. Similar to the information retrieval-based approaches, we retrieve a knowledge subgraph for each NLQ. Moreover, our approach uses graph matching similar to GAnswer and GAQA. However, it is independent of users' assistance to arrive at query graphs and extracts smaller knowledge subgraphs for questions. Additionally, we use different similarity metrics to overcome ambiguities in the graph matching step.

5.4 The Proposed Approach

An overview of the proposed approach is presented in Figure 5.1, which consists of four major steps, including (1) knowledge subgraph construction, (2) question graph generation, (3) graph matching and (4) query execution. The first step targets to construct a knowledge subgraph for each question to avoid exploring the whole KG. To do this, we extend our previous research, BiDPPR [24], which enables us to find knowledge subgraphs through a bi-directed propagation process based on PPR. In the next step, we utilise the syntactic structure of the input NLQ and a set of predefined transformation rules to transform the question into an intermediate form named question graph. The third step matches the question graph to the knowledge subgraph to find the query graph. Finally, the query graph is converted to the corresponding SPARQL query to return the answers to the end-user.

Thus, in our work, a NLQ is converted into a question graph, a query graph is derived from the question graph, and the query graph is transformed into a SPARQL query, as shown in Figure 5.1.

5.4.1 Knowledge Subgraph Construction

The knowledge subgraph is constructed in three stages: (1) topic entity identification, (2) neighbourhood retrieval, and (3) knowledge subgraph retrieval.

5.4.1.1 Topic Entity Identification

The task of EL targets to map each entity mentioned in the questions to its corresponding entity in the KG.

Definition 3 A topic mention is a part of a given NLQ referring to an entity from a KG and shows the main focus of the NLQ. Also, the entity

Figure 5.1: Overview of the proposed approach.

corresponding to the topic mention is called the topic entity. The topic mention and topic entity are denoted as T_m and T_e, respectively.

In this research study, DBpedia spotlight (DBS) [223] is adopted to recognize topic entities for questions.

Definition 4 Given a NLQ, there is at least one topic mention T_m to reflect the main focus of the question.

Definition 5 The entity mention and topic entity of the question "Where is Syngman Rhee buried?" are "Syngman Rhee" and "`http://dbpedia.org/resource/Syngman_Rhee`", respectively.

5.4.1.2 Neighbourhood Retrieval

After identifying the topic entity of a NLQ, all the entities in the KG with the maximum distance m in KG, starting from the topic entity, are extracted (m equals the maximum path length in the question graph, which is discussed in Section 5.4.2).

Definition 6 A neighbourhood graph is defined as $G_n = (N_N, E_N)$ where N_N is a set of entities around the topic entity T_e with distance $d <= m$ from T_e, E_N is a set of edges with distance $d < m$ from T_e, and m is the longest path in the question graph.

5.4.1.3 Knowledge Subgraph

The BiDPPR, a bi-directed propagation technique inspired by PPR [97] and the directed propagation method in [7], is summarized as following:

- A personalized vector which biases towards the topic entity is defined according to the neighbourhood graph G_n. The entries of the personalized vector are set to $\frac{1}{|N_N|}$ for all non-topic entities, and

the corresponding entries of topic entities are set to $1 + \frac{1}{|N_N|}$ where $|N_N|$ is the number of entities in G_n.

$$pr_v^{(0)} = \begin{cases} \frac{1}{|N_N|} + 1 & for\, topic\, entities \\ \frac{1}{|N_N|} & otherwise \end{cases} \tag{5.4.1}$$

- The pre-trained word embeddings BERT[234] is applied to generate the embedding of the question and edges' surface-forms[24] and calculate the weights of the edges. The cosine similarity between the embeddings is considered as the edge's weight.

- The adjacency matrix of the neighbourhood graph G_N is generated according to the edge weights. If M denotes the adjacency matrix, the transpose of M is considered as M^T, including the inverse of relations between entities. The BiDPPR calculation is presented as:

$$\begin{aligned} pr^{(t)} = (1-\alpha)pr^{(t-1)} + \alpha(& \\ & \omega_1 \sum_r \sum_{<n,r,v>} w_r \cdot pr_n^{(t-1)} + \\ & \omega_2 \sum_r \sum_{<v,r,n>} w_r^{(t)} \cdot pr_n^{(t-1)}) \end{aligned} \tag{5.4.2}$$

Where w_r and w_r^t denote the weights of the edge r in both directions based on the matrices M and M^T. Also, ω_1 and ω_2 are assumed as the coefficient ratio for the incoming edges and outgoing edges, respectively.

After T iterations, the top-k entities by BiDPPR score and any edges between them are selected as the relevant facts to shape the question knowledge subgraph. We give a formal definition of knowledge subgraph in Definition 7 as follows:

Definition 7 A knowledge subgraph is defined as $G_k = (N_K, E_K)$ where $N_K \subset N_N$ and $E_K \subset E_N$ and N_K includes the entities with the highest BiDPPR scores which are more likely to be answer entities.

5.4.2 Question Graph Construction

This section discusses how to generate question graphs as intermediate forms to bridge the gap between given question sentences and SPARQL queries.

Definition 8 Given a NLQ q, a question graph is defined as an undirected labeled graph $G_q = (N_q, E_q)$ to present the intention of q where N_q and E_q denote the nodes and edges.

5.4.2.1 Node Recognition

For each input question, we extract three kinds of question nodes, including (1) variable nodes, (2) constant nodes and (3) type nodes.

The constant nodes are basically the named entities mentioned in a question sentence. The variable nodes include the main variable node, which refers to the WH-word of the question, and zero or several middle-variable nodes inserted based on the transformation rules once the question graph is being constructed. To define the type nodes, the noun phrases of the question sentence are extracted. Then, we use a mention-class dictionary to map the noun phrases into the classes of the schema of the underlying KG. Each noun phrase which includes a corresponding class, is assumed as a type node. Those noun phrases not mapped to the classes are later used to shape relations between the question nodes.

A mention-class dictionary consists of all the classes used in the underlying KG along with their hypernyms and synonyms. We use NLTK WordNet[3] to find hypernyms and synonyms of each class.

[3] `https://www.nltk.org/_modules/nltk/corpus/reader/wordnet`

Definition 9 The question sentence "Which books by Kerouac were published by Viking Press?" includes "Kerouac" and "Viking Press" as the constant nodes, "Which" and "book" as the variable node and type node, respectively.

5.4.2.2 Syntactic Parsing

Syntactic parsing (or dependency parsing) is the task of assigning a syntactic structure to a sentence, and a dependency tree is the most widely used syntactic structure. We define the following transformation rules to prune and simplify the generated dependency tree:

- The unnecessary dependency relations (e.g., passive auxiliary, open clausal complement) are filtered to transform the dependence tree to a more simplified syntactic structure [250].
- If a sub-tree only includes the leaf proper-noun-phrase (NNP) nodes, the sub-tree is reduced to a merged node, including the combination of sub-tree's nodes.

  Definition 10 Given the question "When did the Boston Tea Party happen?", the nodes "Boston", "Tea", and "Party" are merged into the "Boston Tea Party" node as shown in Figure 5.2.

5.4.2.3 Question Graph

After pruning the dependency tree, the question graph is created. The nodes of a question graph are constituted from question nodes, and edges are formed based on the transformation rules as follows.

- When a type node is connected to the main variable node in a dependency tree, all the nodes linked to the type node will be connected to the main variable node.

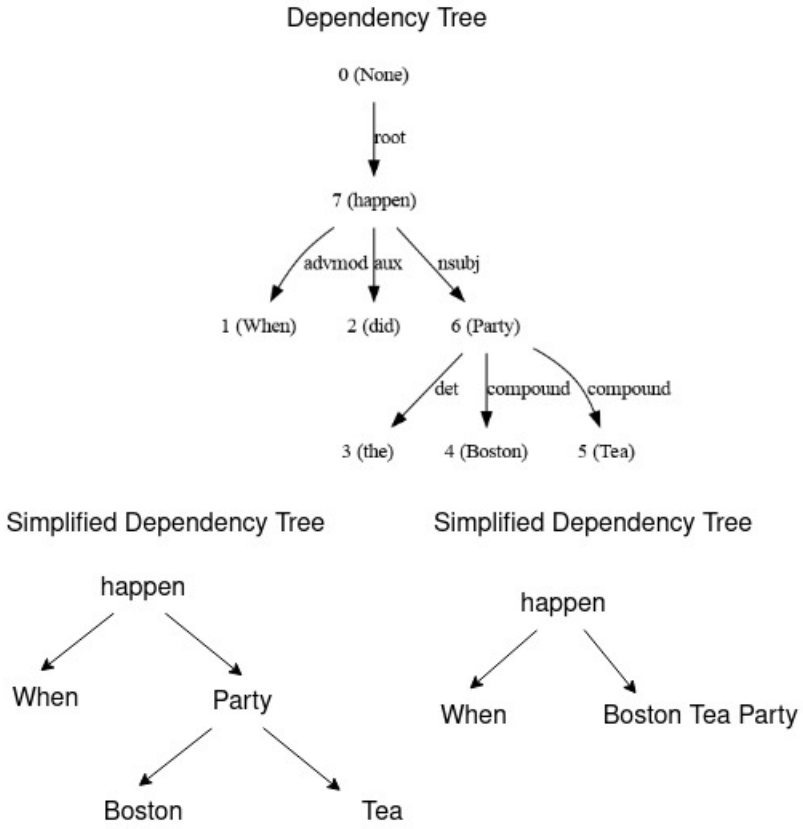


Figure 5.2: Dependency tree and simplified dependency tree of the question "When did the Boston Tea Party happen?".

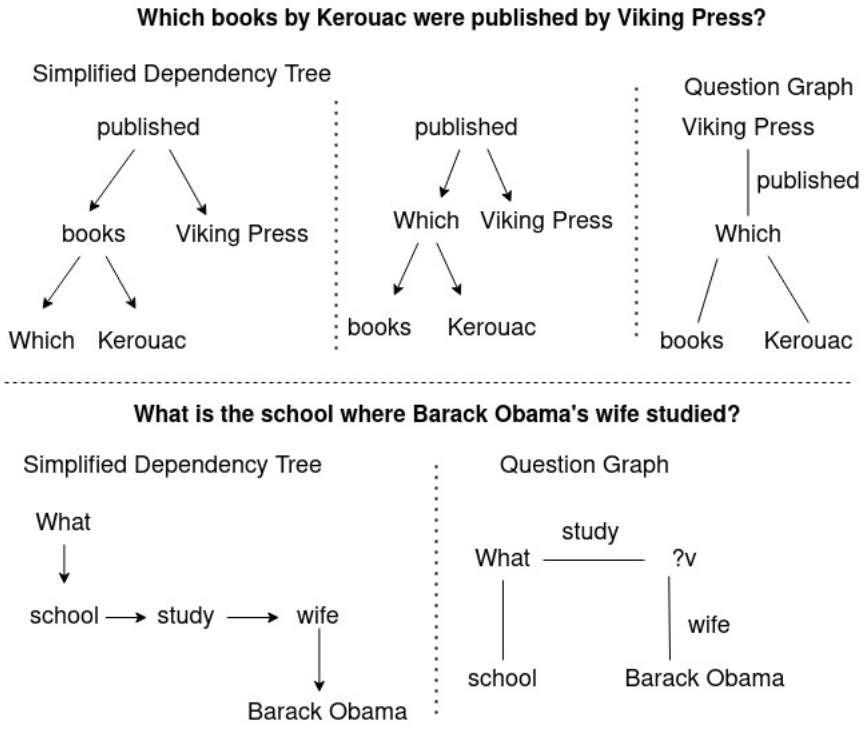


Figure 5.3: Examples of applying the transformation rules.

Definition 11 The type node "books" and main variable node "Which" are connected to each other in the dependency tree of the question "Which books by Kerouac were published by Viking Press?", thus the nodes "Kerouac" and "published" will be linked to the node "Which" as shown in Figure 5.3.

- Two question nodes are linked through an edge in a question graph if the simple path between those two nodes in the dependency tree does not include any other question node. Additionally, the words along the simple path shape the relation [207].

  Definition 12 Since there is no question node along the simple path between "Which" and "Viking Press", as shown in Figure 5.3, these two nodes are connected through an edge in the question graph, represented by the edge label "published".

- If a constant node is connected to a noun-phrase that is not of type node, then a middle-variable node is inserted. The middle-variable node is connected to the constant node, and the noun-phrase node is transformed to the edge between two nodes.

  Definition 13 Given the question "What is the school where Barack Obama's wife studied?", the constant node "Barack Obama" and the noun-phrase "wife" are transformed into the relation "wife" with the connected nodes "Barack Obama" and middle-variable node ?v as depicted in Figure 5.3.

In our work, we divide the edges' labels in the question graphs into three groups, including (1) the edges with known labels, (2) the edges with unknown labels, and (3) the edges describing rdf:type.

Definition 14 As shown in Figure 5.3, the edge between nodes "Which" and "Viking Press" includes the known label "published", and the edge

between nodes "Which" and "Kerouac" has an unknown label. Additionally, the edge between nodes "What" and "school" shows the node "What" is a type (an instance) of node "School" (rdf:type).

Since a question graph has to be matched over a KG, we compute the maximum path length in the question graph and then consider it as the number of required hops m to retrieve the neighbourhood graph (more details in Section 5.4.1.2).

Definition 15 The maximum path length in the G_q is considered as the number of required hops m to build the neighbourhood graph G_n.

5.4.3 Graph Matching

Graph matching is the task of finding a similarity between two graphs through exact or inexact graph matching. The exact graph matching is restricted to finding a one-to-one mapping between two graphs with the name number of nodes, while the inexact graph matching is applied to the cases where the number of nodes is different[251, 252]. Thus, the task of finding the similarity between the question graph and knowledge subgraph is an inexact graph matching problem.

In our work, the directed graph found by node matching and edge matching is named the query graph[4].

Definition 16 A query graph is defined as $G_{qry} = (N_{qry}, E_{qry})$ where is found based on inexact graph matching between the question graph $G_q = (N_q, E_q)$ and knowledge subgraph $G_k = (N_K, E_K)$.

Initially, a query graph includes a topic entity associated with the topic mention and then extends in I iterations. The query graph is expanded in each iteration by matching one more edge or node from the

[4]The nodes and edges of a question graph are spans of the question while the nodes and edges of the query graph are the entities or properties from

question graph with the knowledge subgraph. Thus, the first matching subgraph which is formed in the iteration process is considered as the query graph. The iterative process is terminated when edge matching and node matching are performed for all of the edges and nodes in the question graph. Since unknown edges and variable nodes can be associated with any nodes and edges in the knowledge subgraph, edge matching, and node matching are not performed in these cases.

5.4.3.1 Edge Matching

For each edge in G_q, structural and semantic similarities are utilised to identify its associated edge in G_k [238]. We compute the structural similarity of an edge-pair, and if two edges are similar in the structure, the semantic similarity of the edges is computed.

Definition 17 An edge-pair is denoted as (r_q, r_k) consisting of two edges r_q and r_k where $r_q \in G_q$ and $r_k \in G_k$.

Structural similarity is defined according to the distance of edges from the topic mention T_m and topic entity T_e in G_k and G_q, respectively.

Definition 18 An edge r_q in G_q is structurally similar to an edge r_k in G_k, if the length of simple path between r_q and T_m is equal to the length of simple path between r_k and T_e.

Given an edge-pair (r_q, r_k), semantic similarity is computed if r_q and r_k are in the same distance. Semantic similarity is associated with the similarity between edges based on the likeness of their meanings. We use the edges' surface-forms in the knowledge subgraph and the edges' labels in the question graph to compute semantic similarities. Rather than applying one measure for semantic similarity, we explore the idea of combining existing measures into a semantic similarity ensemble. We take advantage of word embedding BERT, WordNet [253], and Levenstshtein

distance [232]. Given an edge-pair, the embeddings of the surface-form and label are generated using the pre-trained word embedding BERT, and then their cosine similarity is calculated as a similarity score. WordNet is applied to generate two sets of synonyms according to the surface-form and label. Then, we compute the Levenstshtein distance of the elements among two sets, and normalize them as another score of similarity. Algorithm 2 presents the edge matching process.

5.4.3.2 Node Matching

Similar to the edge matching, we use the structural and semantic similarities to map each node in G_q to a corresponding node in G_k.

Thus, given each non-variable node in G_q, the nodes with a similar structure are found in G_k, and then the semantic similarity of those node-pairs is calculated to select the best match in G_q. Note that the applied algorithm for node matching is similar to the edge matching process in Algorithm 2.

5.4.4 Query Execution

Given a query graph, the SPARQL query is constructed through simple conversion rules and then executed over the KG. The primary conversion rules include (1) each edge with its nodes connected presents a triple in which the head node and tail node are the subject and object of the triple, (2) the nodes or edges with the URI resources or literal values are inserted into the triples with the URIs or literal values, (3) the variable nodes or unknown-labeled edges are inserted with the variable names, and (4) the main variable (WH-node) which is the answer of the factoid question, needs to be fetched in the SELECT part of the query.

We apply a set of simple rules to handle questions, including aggregation phrases. We assume there are only a few aggregation phrases

(how many, how much, greater than, less than, and etc) and then define a set of simple rules to add constraints to the queries. For example, in the question "How many moons does Mars have?", the COUNT rule is used to count the main variable.

Algorithm 2 Edge Matching Process

Input: Question graph G_q, Knowledge subgraph G_k

Result: Vector o including the matched edges from G_k

for *each edge* e_k *in* G_k **do**
  Compute d_k as the length of simple path between e_k and topic entity T_e Find S_{ek} as the set of synonyms for e_k
end
for *each edge* e_q *in* G_q **do**
  if e_q *has a known label* **then**
    Find S_{eq} as the set of synonyms for e_q **for** *each edge* e_k *in* G_k **do**
      if d_k *equals* d_q **then**
        Compute s_{cs} as the cosine similarity between the embeddings of e_q and e_k
        Compute and normalize the Levenstshtein distance between elements of sets S_{ek} and S_{eq}, and select the maximum as s_{ld}
        Calculate S_{sim} as the liner combination of s_{ld} and s_{cs}
        Store S_{sim} as the similarity score of the edge-pair (e_q, e_k) in vector v
      end
    end
    Find (e_q, m_k) with the highest S_{sim} from v Store m_k as the matched edge of e_q in vector o
  end
end

Return vector o

5.5 Experimental Study

The proposed approach is evaluated on DBPedia[22] as the KG using the questions of QALD-6 as the QA dataset. The results are compared with

all systems in the QALD-6 competition as well as GAQA and GAnswer.

5.5.1 Dataset

KG In our experiment, DBPedia as an open-domain large-scaled KG is adopted. DBPedia is extracted from structured data in Wikipedia through a crowd sourcing community.

QA Benchmark The Question Answering over Linked Data challenge (QALD) provides up-to-date benchmarks for assessing and comparing QASs over DBPedia. The QALD-6 which is the sixth installment of the QALD challenge, includes 350 training questions and 100 test questions.

5.5.2 Metrics

We adopt three evaluation metrics, including recall, precision and F1-score, formulated as the following:

$$\text{Recall} = \frac{\text{correctly retrieved answers}}{\text{gold standard answers}} \tag{5.5.1}$$

$$\text{Precision} = \frac{\text{correctly obtained answers}}{\text{returned answers}} \tag{5.5.2}$$

$$\text{F1} = 2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}} \tag{5.5.3}$$

5.5.3 Results and Discussion

Table 5.1 presents the experimental results, where 'Right' denotes the number of questions that are correctly answered. Our proposed approach, denoted GDQA (Graph Driven Question Answering), achieves 0.75%, 0.82%, and 0.78% for the recall, precision, and F1, respectively and answers 72 questions correctly, outperforming the participants in the

QALD-6 competitions. Compared to GAnswer (NFF and RFF), QDQA improves the recall by 5% and correctly answers more questions by 4%.

Although the results of GAQA and CANaLI[254] show slightly better performance than our results, GAQA requires users to submit questions as question graphs (see Section 5.3) and CANaLI uses a controlled natural language (CNL) to restrict the grammar used to pose input questions.

Table 5.1: Main results on QALD-6 test set.

	Processed	Right	Recall	Precision	F1
GDQA	100	72	0.75	0.82	0.78
GAQA	100	74	0.83	0.87	0.84
NFF	100	68	0.70	0.89	0.78
RFF	100	40	0.43	0.77	0.55
CANaLI	100	83	0.89	0.89	0.89
UTQA	100	63	0.69	0.82	0.75
KWGAnswer	100	52	0.59	0.85	0.70
SemGraphQA	100	20	0.25	0.70	0.37
UIQA1	44	21	0.63	0.54	0.25
UIQA2	36	14	0.53	0.43	0.17

According to 28% questions that our proposed approach can not answer correctly, the failure analysis of the work is provided. We find that there are two main reasons for the failure to answer questions. The first failure reason is question graph construction. As an example, the question graph of the question "How many grand-children did Jacques Cousteau have?" includes "How many" and "Jacques Cousteau" as the variable node and constant node, respectively and "grand-children" as the relation while the answer entities are accessed through the question graph: {<How many, child, ?v>, <?v, child, Jacques Cousteau>}. The second reason is finding semantic similar nodes or edges in the graph matching step. For example, in the question "Where is Syngman Rhee buried?", the relation "burn" in the question graph has to match the relation "restingPlace" in the knowledge subgraph while the edge matching is failed to find the matching. Since the other reasons, such as knowledge subgraph construction, and complex aggregation, happen far much less

than the first two reasons, their ratio is reported in total.

From this experiment, we find out that our approach achieves high performance in answering questions over KGs. While our approach is tested on an open-domain KG, it can be generalised across diverse KGs, which we regard as an advantage of the work. In addition, end-users are not restricted to posing their questions in a specific format (e.g., graphs or CNL).

We also acknowledge the following limitations of the work. We assume that every question includes at least one topic mention, whereas there may be no topic mention in some cases (e.g., "Who died in the same place where they were born?"). Moreover, although our approach can answer some aggregation questions (e.g., count questions), it may not answer questions with more complex aggregation functions.

5.6 Conclusions

QA over KGs provides an easy and effective way for end-users to access KGs using NLQs rather than complicated structured query languages such as SPARQL. We propose a graph-driven approach to answer NLQs in four steps: (1) knowledge subgraph construction, (2) question graph construction, (3) matching graphs, and (4) query execution. Our experimental results show that the proposed approach achieves significantly higher recall, precision and F1-score in comparison to the QALD competition's results; however, there still exists much space for improvement of QA over KGs.

A possible direction for our future work is to utilise semantic parsing as well as syntactic parsing in the question graph generation step and address the limitations, which are highlighted in Section 5.5.3.

Acknowledgment

This work has been partially funded by the project WordLiftNG within the Eureka, Eurostars Programme (grant agreement number 877857 with the Austrian Research Promotion Agency (FFG)) and the project KI-Net within the Interreg Österreich-Bayern 2014-2020 programme (grant agreement number AB 292). Additionally, we would like to express our gratitude to Oleksandra Roche-Newton for providing feedback that aided in the manuscript's improvement.

List of Acronyms

Notation	Description
ACM	Association of Computing Machinery.
AI	Artificial Intelligence.
API	Application Programming Interface.
BERT	Bidirectional Encoder Representations from Transformers.
BiDPPR	Bi-Directed Personal Page Rank.
CNL	Controlled Natural Language.
CNN	Convolutional Neural Network.
CSV	Comma-Separated Values.
CVT	Compound Value Type.
DBLP	Digital Bibliography & Library Project.
DM	Dialog Management.
EB	Exabytes.
EL	Entity Linking.
FAQ	Frequently Asked Question and answer.
FTP	File Transfer Protocol.
GAnswer	Graph Answer.
GBDT	Gradient Boosted Decision Tree.
GRU	Gated Recurrent Unit.
HITS	Hyperlink-Induced Topic Search.
HRED	Hierarchical Recurrent Encoder-Decoder.

Notation	Description
HTML	HyperText Markup Language.
IEEE	Institute of Electrical and Electronics Engineers.
IR	Information Retrieval.
JSON-LD	JavaScript Object Notation for Linked Data.
KB	Knowledge Base.
KG	Knowledge Graphs.
KGQAS	Knowledge Graph-based Question Answering System.
LR	Learning Rate.
LSTM	Long Short-Term Memory.
MB	Megabytes.
ML	Machine Learning.
MLP	Multi-Layer Perceptron.
NER	Named Entity Recognition.
NFF	Node-First Framework.
NLP	Natural Language Processing.
NLQ	Natural Language Question.
NLTK	Natural Language Toolkit.
NLU	Natural Language Understanding.
NNP	Proper-Noun-Phrase.
OWL	Web Ontology Language.
POS	Part Of Speech.
PP	Page Rank.
PPR	Personal Page Rank.
PRN	PageRank-Nibble.

Notation	Description
QA	Question Answering.
QALD-6	The sixth installment of the Question Answering over Linked Data challenge.
QAS	Question Answering System.
RDF	Resource Description Framework.
RDFa	Resource Description Framework in Attributes.
RDFs	Resource Description Framework Schema.
REST	Representational State Transfer.
RFF	Relation-First Framework.
RLU	Rectified Linear Unit.
RNN	Recurrent Neural Network.
RSS	Really Simple Syndication.
SME	Small and Medium-sized Enterprise.
SBERT	Sentence-Bidirectional Encoder Representations from Transformers.
SPARQL	Simple Protocol and RDF Query Language.
SQL	Structured Query Language.
SVM	Support Vector Machine.
SWSE	SWSE.
URI	Uniform Resource Identifiers.
URL	Uniform Resource Locator.
W3C	World Wide Web Consortium.
WASA	Schema.org Actions.
WebQSP	WebQuestionsSP.
XGBoost	Extreme Gradient Boost.
XML	Extensible Markup Language.

Notation	Description
GAQA	Graph Alignment Question Answering.

List of Figures

List of Tables

Thesis Publications

[1] **Sareh Aghaei**, Elie Raad, and Anna Fensel. "Question Answering Over Knowledge Graphs: A Case Study in Tourism". In: *IEEE Access* Vol. 10 (2022), pp. 69788–69801. DOI: 10.1109/ACCESS.2022.3187178.

[2] **Sareh Aghaei**, Kevin Angele, and Anna Fensel. "Building knowledge subgraphs in question answering over knowledge graphs". In: *Proceedings of the 22nd International Conference on Web Engineering*. Bari, Italy: Springer-Verlag, 2022, pp. 237–251. DOI: 10.1007/978-3-031-09917-5_16.

[3] **Sareh Aghaei**, Kevin Angele, Elwin Huaman, Geni Bushati, Mathias Schiestl, and Anna Fensel. "Interactive Search on the Web: The Story So Far". In: *Information* Vol. 13, No. 7 (2022), p. 324. DOI: 10.3390/info13070324.

[4] **Sareh Aghaei** and Anna Fensel. "Finding Similar Entities across Knowledge Graphs". In: *Proceedings of the 7th International Conference on Advances in Computer Science and Information Technology*. Vienna, Austria, 2021, pp. 1–11. DOI: 10.5121/csit.2021.110301.

[5] **Sareh Aghaei**. "Question Answering over Knowledge Graphs". In: *Proceedings of the 5th Doctoral Consortium at Rules and Reasoning (RuleML+RR)*. Vol. 2956. Leuven, Belgium: CEUR-WS.org, 2021. URL: http://ceur-ws.org/Vol-2956/paper35.pdf.

[6] **Sareh Aghaei**, Sepide Masoudi, Tek Raj Chhetri, and Anna Fensel. "Question Answering over Knowledge Graphs: A Graph-Driven Approach". In: *Proceedings of the 21st IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*. forthcoming. Niagara Falls, Canada, 2022.

[7] Amar Tauqeer, Ismaheel Hammid, **Sareh Aghaei**, Parvaneh Parvin, Elbrich M. Postma, and Fensel Anna. "Smell and Taste Disorders Knowledge Graph: Answering Questions Using Health Data". In: *Expert Systems with Applications* (2022). under review.

[8] Tek Raj Chhetri, **Sareh Aghaei**, Anna Fensel, Ulrich Göhner, Sebnem Gül-Ficici, and Jorge Martinez-Gil. "Optimising Manufacturing Process with Bayesian Structure Learning and Knowledge Graphs". In: *Proceedings of the 18th International Conference on Computer Aided Systems Theory*. Las Palmas de Gran Canaria, Spain, 2023, pp. 1–9. DOI: 10.1007/978-3-031-25312-6_70.

Bibliography

[1] Peiyun Wu, Xiaowang Zhang, and Zhiyong Feng. "A Survey of Question Answering over Knowledge Base". In: *Proceedings of China Conference on Knowledge Graph and Semantic Computing*. 2019, pp. 86–97. DOI: 10.1007/978-981-15-1956-7_8.

[2] Michael Caballero. "A Brief Survey of Question Answering Systems". In: *International Journal of Artificial Intelligence & Applications (IJAIA)* Vol. 12, No. 5 (2021). DOI: 10.5121/ijaia.2021.12501.

[3] Mohammad Yani and Adila Alfa Krisnadhi. "Challenges, Techniques, and Trends of Simple Knowledge Graph Question Answering: A Survey". In: *Information* Vol. 12, No. 7 (2021), p. 271. DOI: 10.3390/info12070271.

[4] Weiguo Zheng, Lei Zou, Xiang Lian, Jeffrey Xu Yu, Shaoxu Song, and Dongyan Zhao. "How to build templates for RDF question/answering: An uncertain graph similarity join approach". In: *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*. 2015, pp. 1809–1824. ISBN: 9781450327589. DOI: 10.1145/2723372.2747648.

[5] Abdalghani Abujabal, Rishiraj Saha Roy, Mohamed Yahya, and Gerhard Weikum. "Never-ending learning for open-domain question answering over knowledge bases". In: *Proceedings of the 2018 World Wide Web Conference*. 2018, pp. 1053–1062. DOI: 10.1145/3178876.3186004.

[6] Abdalghani Abujabal, Mohamed Yahya, Mirek Riedewald, and Gerhard Weikum. "Automated template generation for question answering over knowledge graphs". In: *Proceedings of the 26th International Conference on World Wide Web*. 2017, pp. 1191–1200. ISBN: 9781450349130. DOI: 10.1145/3038912.3052583.

[7] Haitian Sun, Bhuwan Dhingra, Manzil Zaheer, Kathryn Mazaitis, Ruslan Salakhutdinov, and William Cohen. "Open Domain Question Answering Using Early Fusion of Knowledge Bases and Text". In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. 2018, pp. 4231–4242. DOI: 10.18653/v1/D18-1455.

[8] Haitian Sun, Tania Bedrax Weiss, and William Cohen. "PullNet: Open Domain Question Answering with Iterative Retrieval on Knowledge Bases and Text". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*. Vol. abs/1904.09537. 2019, pp. 2380–2390. DOI: 10.18653/v1/D19-1242.

[9] Gaole He, Yunshi Lan, Jing Jiang, Wayne Xin Zhao, and Ji Rong Wen. "Improving multi-hop knowledge base question answering by learning intermediate supervision signals". In: *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*. 2021, pp. 553–561. DOI: 10.1145/3437963.3441753.

[10] Bin Fu, Yunqi Qiu, Chengguang Tang, Yang Li, Haiyang Yu, and Jian Sun. "A survey on complex question answering over knowledge base: Recent advances and challenges". In: *ArXiv* (2020). DOI: 10.48550/arXiv.2007.13069.

[11] Weiguo Zheng, Hong Cheng, Jeffrey Xu Yu, Lei Zou, and Kangfei Zhao. "Interactive natural language question answering over knowledge graphs". In: *Information Sciences* Vol. 481 (2019), pp. 141–159. DOI: 10.1016/j.ins.2018.12.032.

[12] Kun Xu, Lingfei Wu, Zhiguo Wang, Mo Yu, Liwei Chen, and Vadim Sheinin. "Exploiting rich syntactic information for semantic parsing with graph-to-sequence model". In: *ArXiv* (2018). DOI: 10.48550/arXiv.1808.07624.

[13] Sincy Thambi and Reghu Raj. "Towards Improving the Performance of Question Answering System using Knowledge Graph-A Survey". In: *Proceedings of the 2022 2nd International Conference on Artificial Intelligence and Smart Energy (ICAIS)*. IEEE. 2022, pp. 672–679. DOI: 10.1109/ICAIS53314.2022.9742802.

[14] Dennis Diefenbach. "Question answering over Knowledge Bases". Accessed: 2022-12-16. PhD dissertation. Université de Lyon, 2018. URL: https://theses.hal.science/tel-02497232.

[15] Suneera Cm, Jay Prakash, and Pramod Kumar Singh. *Question Answering Over Knowledge Graphs Using Bert Based Relation Mapping*. Accessed: 2022-12-16. 2022. URL: https://ssrn.com/abstract=4102838.

[16] *Graph Technology at your fingertips*. Accessed: 2023-01-23. URL: https://www.wvr.io/.

[17] *WordLift New Generation*. Accessed: 2022-04-10. URL: https://wordlift.io/ng/.

[18] *KI-NET: AI-based optimization in production*. Accessed: 2022-04-10. URL: https://scch.at/en/das-projects-details/ki-net.

[19] Ram Ganesan Athreya. "Template-Based Question Answering over Linked Data using Recursive Neural Networks". Accessed: 2022-12-16. MSc thesis. Arizona State University, 2018. URL: http://core.ac.uk/download/pdf/195379916.pdf.

[20] Hamid Zafartavanaelmi. "Semantic Question Answering Over Knowledge Graphs: Pitfalls and Pearls". Accessed: 2022-12-16. PhD dissertation. Universitäts-und Landesbibliothek Bonn, 2021. URL: https://bonndoc.ulb.uni-bonn.de/xmlui/handle/20.500.11811/9125.

[21] Alaa Mohasseb, Mohamed Bader El Den, and Mihaela Cocea. "Classification of factoid questions intent using grammatical features". In: *ICT Express* Vol. 4, No. 4 (2018). Accessed: 2022-12-16, pp. 239–242. ISSN: 2405-9595. DOI: 10.1016/j.icte.2018.10.004.

[22] Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick Van Kleef, Sören Auer, and Christian Bizer. "Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia". In: *Semantic web* Vol. 6, No. 2 (2015), pp. 167–195. DOI: 10.3233/SW-140134.

[23] Philipp Christmann, Rishiraj Saha Roy, and Gerhard Weikum. "Beyond NED: Fast and Effective Search Space Reduction for Complex Question Answering over Knowledge Bases". In: *Proceedings of the 15th ACM International Conference on Web Search and Data Mining*. Association for Computing Machinery, 2022, pp. 172–180. ISBN: 9781450391320. DOI: 10.1145/3488560.3498488.

[24] Sareh Aghaei and Anna Fensel. "Building knowledge subgraphs in question answering over knowledge graphs". In: vol. 13362. Springer-Verlag, 2022, pp. 237–251. ISBN: 978-3-031-09916-8. DOI: 10.1007/978-3-031-09917-5_16.

[25] Taher Haveliwala. "Topic-sensitive PageRank". In: *World Wide Web* (2002), pp. 517–526. DOI: 10.1145/511446.511513.

[26] Erol Bozsak, Marc Ehrig, Siegfried Handschuh, Andreas Hotho, Alexander Maedche, Boris Motik, Daniel Oberle, Christoph Schmitz, Steffen Staab, and Ljiljana Stojanovic. "KAON—Towards a large scale semantic web". In: *International Conference on Electronic Commerce and Web Technologies*. Springer. 2002, pp. 304–313. DOI: 10.1007/3-540-45705-4_32.

[27] Marta Sabou. "An Introduction to Semantic Web Technologies". In: *Semantic Web Technologies for Intelligent Engineering Applications*. Springer International Publishing, 2016, pp. 53–81. ISBN: 978-3-319-41490-4. DOI: 10.1007/978-3-319-41490-4_3.

[28] Marta Sabou, Fajar Juang Ekaputra, and Stefan Biffl. "Semantic Web Technologies for Data Integration in Multi-Disciplinary Engineering". In: *Multi-Disciplinary Engineering for Cyber-Physical Production Systems: Data Models and Software Solutions for Handling Complex Engineering Projects*. Springer International Publishing, 2017, pp. 301–329. ISBN: 978-3-319-56345-9. DOI: 10.1007/978-3-319-56345-9_12.

[29] Ora Lassila and Ralph Swick. *Resource description framework (RDF) model and syntax specification*. Accessed: 2022-12-16. Citeseer, 1998. URL: http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/.

[30] Elias Kärle. "Semantic Data Publication and Consumption". Accessed: 2022-12-16. PhD dissertation. Universität Innsbruck, 2021. URL: https://diglib.uibk.ac.at/ulbtirolhs/content/titleinfo/6676633/full.pdf.

[31] Dieter Fensel, Umutcan Şimşek, Kevin Angele, Elwin Huaman, Elias Kärle, Oleksandra Panasiuk, Ioan Toma, Jürgen Umbrich, and Alexander Wahler. "Introduction: what is a knowledge graph?" In: *Knowledge Graphs*. Springer, 2020, pp. 1–10. DOI: 10.1007/978-3-030-37439-6_1.

[32] Tobias Weller. *Learning Latent Features using Stochastic Neural Networks on Graph Structured Data*. PhD dissertation. Accessed: 2022-12-16. 2021. URL: https://d-nb.info/1230475656/34.

[33] Heiko Paulheim. "Knowledge graph refinement: A survey of approaches and evaluation methods". In: *Semantic web* Vol. 8, No. 3 (2017), pp. 489–508. DOI: 10.3233/SW-160218.

[34] Tobias Weller and Maribel Acosta. "Predicting Instance Type Assertions in Knowledge Graphs Using Stochastic Neural Networks". In: *Proceedings of the 30th ACM International Conference on Information and Knowledge Management*. 2021, pp. 2111–2118. ISBN: 9781450384469. DOI: 10.1145/3459637.3482377.

[35] Muhammad Yahya, John Breslin, and Muhammad Intizar Ali. "Semantic web and knowledge graphs for industry 4.0". In: *Applied Sciences* Vol. 11, No. 11 (2021), p. 5110. DOI: 10.3390/app11115110.

[36] Denny Vrandečić and Markus Krötzsch. "Wikidata: a free collaborative knowledgebase". In: *Communications of the ACM* Vol. 57, No. 10 (2014), pp. 78–85. DOI: 10.1145/2629489.

[37] Fabian Suchanek, Gjergji Kasneci, and Gerhard Weikum. "Yago: A Core of Semantic Knowledge". In: *Proceedings of the 16th International Conference on World Wide Web*. WWW '07. Association for Computing Machinery, 2007, pp. 697–706. ISBN: 9781595936547. DOI: 10.1145/1242572.1242667.

[38] Xin Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. "Knowledge vault: A web-scale approach to probabilistic knowledge fusion". In: *Proceedings of the 20th ACM Special Interest Group on Knowledge Discovery and Data Mining (SIGKDD) International Conference on Knowledge Discovery and Data Mining*. 2014, pp. 601–610. DOI: 10.1145/2623330.2623623.

[39] Andrew Carlson, Justin Betteridge, Richard Wang, Estevam Hruschka Jr, and Tom Mitchell. "Coupled semi-supervised learning for information extraction". In: *Proceedings of the 3rd ACM International Conference on Web Search and Data Mining*. 2010, pp. 101–110. DOI: 10.1145/1718487.1718501.

[40] Zaenal Akbar, Anna Fensel, and Dieter Fensel. "On Generating Stories from Semantically Annotated Tourism-Related Content". In: *Proceedings of the OTM Confederated International Conferences On the Move to Meaningful Internet Systems*. Springer. 2018, pp. 481–497. DOI: 10.1007/978-3-030-02671-4_29.

[41] Andreas Lothe Opdahl, Tareq Al Moslmi, Duc Tien Dang Nguyen, Marc Gallofré Ocaña, Bjørnar Tessem, and Csaba Veres. "Semantic Knowledge Graphs for the News: A Review". In: *ACM Computing Surveys* Vol. 55, No. 7 (2022). ISSN: 0360-0300. DOI: 10.1145/3543508.

[42] Maya Rotmensch, Yoni Halpern, Abdulhakim Tlimat, Steven Horng, and David Sontag. "Learning a health knowledge graph from electronic medical records". In: *Scientific reports* Vol. 7, No. 1 (2017), pp. 1–11. DOI: 10.1038/s41598-017-05778-z.

[43] Elem Güzel Kalaycı, Irlan Grangel González, Felix Lösch, Guohui Xiao, Evgeny Kharlamov, and Diego Calvanese. "Semantic integration of Bosch manufacturing data using virtual knowledge graphs". In: *Proceedings of the International Semantic Web Conference*. Springer. 2020, pp. 464–481.

[44] Michel Oleynik. *Leveraging Word Embedding for Biomedical Natural Language Processing*. PhD dissertation. Accessed: 2022-12-16. 2020. URL: https://online.medunigraz.at/mug_online/wbabs.getDocument?pThesisNr=58784&pAutorNr=91542&pOrgNR=1.

[45] Ethem Alpaydin. "Introduction to machine learning (adaptive computation and machine learning series)". In: *The MIT Press Cambridge* (2004).

[46] Tom Mitchell. *Machine learning*. Vol. 1. No. 9. McGraw-hill New York, 1997.

[47] Liang Zhang, Jin Wen, Yanfei Li, Jianli Chen, Yunyang Ye, Yangyang Fu, and William Livingood. "A review of machine learning in building load prediction". In: *Applied Energy* Vol. 285 (2021), p. 116452. DOI: 10.1016/j.apenergy.2021.116452.

[48] Thorsten Wuest, Daniel Weimer, Christopher Irgens, and Klaus Dieter Thoben. "Machine learning in manufacturing: advantages, challenges, and applications". In: *Production & Manufacturing Research* Vol. 4 (2016), pp. 23–45. DOI: 10.1080/21693277.2016.1192517.

[49] Wang Bao, Ning Lianju, and Kong Yue. "Integration of unsupervised and supervised machine learning algorithms for credit risk assessment". In: *Expert Systems with Applications* Vol. 128 (2019), pp. 301–315. ISSN: 0957-4174. DOI: doi.org/10.1016/j.eswa.2019.02.033.

[50] Leslie Pack Kaelbling, Michael Littman, and Andrew Moore. "Reinforcement learning: A survey". In: *Journal of Artificial Intelligence Research* Vol. 4 (1996), pp. 237–285. DOI: 10.1613/jair.301.

[51] Anil Kumar Jain, Jianchang Mao, and Moidin Mohiuddin. "Artificial neural networks: A tutorial". In: *Computer* Vol. 29, No. 3 (1996), pp. 31–44. DOI: 10.1109/2.485891.

[52] Christian Janiesch, Patrick Zschech, and Kai Heinrich. "Machine learning and deep learning". In: *Electronic Markets* Vol. 31, No. 3 (2021), pp. 685–695. DOI: 10.1007/s12525-021-00475-2.

[53] Guoqiang Zhang, Butje Eddy Patuwo, and Michael Hu. "Forecasting with artificial neural networks:: The state of the art". In: *International Journal of Forecasting* Vol. 14, No. 1 (1998), pp. 35–62. ISSN: 0169-2070. DOI: 10.1016/S0169-2070(97)00044-7.

[54] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Deep learning". In: *Nature* Vol. 521, No. 7553 (2015). Accessed: 2022-12-16, pp. 436–444. URL: https://www.nature.com/articles/nature14539.

[55] Prakash Nadkarni, Lucila Ohno Machado, and Wendy Chapman. "Natural language processing: an introduction". In: *Journal of the American Medical Informatics Association* Vol. 18, No. 5 (2011), pp. 544–551. ISSN: 1067-5027. DOI: 10.1136/amiajnl-2011-000464.

[56] Julia Hirschberg and Christopher Manning. "Advances in natural language processing Science". In: *Science* Vol. 349, No. 6245 (2015), pp. 261–266. DOI: 10.1126/science.aaa8685.

[57] Italo Oliveira, Renato Fileto, René Speck, Luis Garcia, Diego Moussallem, and Jens Lehmann. "Towards holistic entity linking: Survey and directions". In: *Information Systems* Vol. 95 (2021), p. 101624. DOI: 10.1016/j.is.2020.101624.

[58] Wei Shen, Jianyong Wang, and Jiawei Han. "Entity linking with a knowledge base: Issues, techniques, and solutions". In: *IEEE Transactions on Knowledge and Data Engineering* Vol. 27, No. 2 (2014), pp. 443–460. DOI: 10.1109/TKDE.2014.2327028.

[59] Eric Brill and Raymond Mooney. "An overview of empirical natural language processing". In: *AI magazine* Vol. 18, No. 4 (1997), pp. 13–13. DOI: 10.1609/aimag.v18i4.1318.

[60] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. "Efficient estimation of word representations in vector space". In: *ArXiv* (2013). DOI: 10.48550/arXiv.1301.3781.

[61] Jeffrey Pennington, Richard Socher, and Christopher Manning. "Glove: Global vectors for word representation". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2014, pp. 1532–1543. DOI: 10.3115/v1/D14-1162.

[62] Jacob Devlin, Ming Wei Chang, Kenton Lee, and Kristina Toutanova. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Asso-

ciation for Computational Linguistics, 2019, pp. 4171–4186. DOI: 10.18653/v1/n19-1423.

[63] Abdelghani Bouziane, Djelloul Bouchiha, Noureddine Doumi, and Mimoun Malki. "Question answering systems: survey and trends". In: *Procedia Computer Science* Vol. 73 (2015), pp. 366–375. DOI: 10.1016/j.procs.2015.12.005.

[64] Dennis Diefenbach, Vanessa Lopez, Kamal Singh, and Pierre Maret. "Core techniques of question answering systems over knowledge bases: a survey". In: *Knowledge and Information Systems* Vol. 55 (2017), pp. 529–569. DOI: 10.1007/s10115-017-1100-y.

[65] Bert Green Jr, Alice Wolf, Carol Chomsky, and Kenneth Laughery. "Baseball: an automatic question-answerer". In: *Proceedings of the May 9-11, 1961, Western Joint IRE-AIEE-ACM Computer Conference*. 1961, pp. 219–224. DOI: 10.1145/1460690.1460714.

[66] Bolanle Ojokoh and Emmanuel Adebisi. "A Review of Question Answering Systems". In: *Journal of Web Engineering* Vol. 17 (2019), pp. 717–758. DOI: 10.13052/jwe1540-9589.1785.

[67] Maidong Hu and Abdel Halim Elamy. "Speeding Up QA: An Index Structure for Question Queries". In: *Proceedings of the 2007 Canadian Conference on Electrical and Computer Engineering*. 2007, pp. 1215–1218. DOI: 10.1109/CCECE.2007.310.

[68] *Summary of 2019 Gartner IIoT Platform Report*. Accessed: 2022-6-20. URL: https://medium.com/world-of-iot/96-summary-of-2019-gartner-iiot-platform-report-9d7f41f53a4e.

[69] *Internet of Things: Key stats for 2022*. Accessed: 2022-6-20. URL: https://techinformed.com/internet-of-things-key-stats-for-2022/.

[70] Santha Sheela and Jaya Kumar. "Comparative study of syntactic search engine and semantic search engine: A survey". In: *Proceedings of the 2019 Fifth International Conference on Science Technology Engineering and Mathematics*. Vol. 1. IEEE. 2019, pp. 1–4. DOI: 10.1109/ICONSTEM.2019.8918837.

[71] Artur Strzelecki. "Eye-tracking studies of web search engines: A systematic literature review". In: *Information* Vol. 11, No. 6 (2020), p. 300. DOI: 10.3390/info11060300.

[72] Estela Mariné Roig. "Measuring destination image through travel reviews in search engines". In: *Sustainability* Vol. 9, No. 8 (2017), p. 1425. DOI: 10.3390/su9081425.

[73] Andrei Broder. "A taxonomy of web search". In: *ACM Sigir forum*. Vol. 36. 2. ACM New York, USA. 2002, pp. 3–10.

[74] Bin Fu, Yunqi Qiu, Chengguang Tang, Yang Li, Haiyang Yu, and Jian Sun. "A Survey on Complex Question Answering over Knowledge Base: Recent Advances and Challenges". In: *ArXiv* Vol. abs/2007.13069 (2020). DOI: 10.48550/arXiv.2105.11644.

[75] Hongshen Chen, Xiaorui Liu, Dawei Yin, and Jiliang Tang. "A Survey on Dialogue Systems: Recent Advances and New Frontiers". In: *ACM Special Interest Group on Knowledge Discovery and Data Mining (SIGKDD) Explorations Newsletter* Vol. 19, No. 2 (2017), pp. 25–35. ISSN: 1931-0145. DOI: 10.1145/3166054.3166058.

[76] Shafquat Hussain, Omid Ameri Sianaki, and Nedal Ababneh. "A survey on conversational agents/chatbots classification and design techniques". In: *Workshop Proceedings of the International Conference on Advanced Information Networking and Applications*.

Springer. 2019, pp. 946–956. DOI: 10.1007/978-3-030-15035-8_93.

[77] Daniel Jurafsky and James Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Vol. 2. Accessed: 2022-12-16. 2008. URL: https://web.stanford.edu/~jurafsky/slp3/ed3book.pdf.

[78] Vishal Gupta. "A keyword searching algorithm for search engines". In: *Proceedings of the 2007 Innovations in Information Technologies*. 2007, pp. 203–207. DOI: 10.1109/IIT.2007.4430467.

[79] Christopher Stokoe. "Differentiating homonymy and polysemy in information retrieval". In: *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*. Vancouver, British Columbia, Canada, 2005, pp. 403–410. DOI: 10.3115/1220575.1220626.

[80] Sareh Aghaei, Mohammad Ali Nematbakhsh, and Hadi Khosravi Farsani. "Evolution of the world wide web : from web 1.0 to web 4.0". In: *International Journal of Web and Semantic Technology* Vol. 3 (2012), pp. 1–10. DOI: 10.5121/ijwest.2012.3101.

[81] Paolo Ceravolo, Chengfei Liu, Mustafa Jarrar, and Kai Uwe Sattler. "Special issue on querying the data web". In: *World Wide Web* Vol. 14, No. 5 (2011), pp. 461–463. DOI: 10.1007/s11280-011-0139-z.

[82] Victoria Uren, Yuangui Lei, Vanessa Lopez, Haiming Liu, Enrico Motta, and Marina Giordanino. "The usability of semantic search tools: a review". In: *The Knowledge Engineering Review* Vol. 22 (2007), pp. 361–377. DOI: 10.1017/S0269888907001233.

[83] Ali Mohamed Nabil Allam and Mohamed Hassan Haggag. "The question answering systems: A survey". In: *International Journal of Research and Reviews in Information Sciences* Vol. 2, No. 3 (2012).

[84] Dennis Diefenbach, Andreas Both, Kamal Singh, and Pierre Maret. "Towards a question answering system over the semantic web". In: *Semantic Web* Vol. 11, No. 3 (2020), pp. 421–439. DOI: 10.3233/SW-190343.

[85] Suket Arora, Kamaljeet Batra, and Sarabjit Singh. "Dialogue System: A Brief Review". In: *ArXiv* (2013). DOI: abs/1306.4134.

[86] Yuya Chiba, Takashi Nose, Taketo Kase, Mai Yamanaka, and Akinori Ito. "An analysis of the effect of emotional speech synthesis on non-task-oriented dialogue system". In: *Proceedings of the 19th Annual SIGdial Meeting on Discourse and Dialogue*. 2018, pp. 371–375. DOI: 10.18653/v1/W18-5044.

[87] Umutcan Simsek, Jürgen Umbrich, and Dieter Fensel. "Towards a Knowledge Graph Lifecycle: A pipeline for the population of a commercial Knowledge Graph". In: *Qurator*. Accessed: 2022-12-16. 2020. URL: https://ceur-ws.org/Vol-2535/paper_10.pdf.

[88] Tom Seymour, Dean Frantsvog, and Satheesh Kumar. "History of search engines". In: *International Journal of Management & Information Systems* Vol. 15, No. 4 (2011), pp. 47–58. DOI: 10.19030/ijmis.v15i4.5799.

[89] Sampath Kumar and Saranathan Pavithra. "Evaluating the searching capabilities of search engines and metasearch engines: a comparative study". In: *Annals of Library and Information Studies* Vol. 57 (2010). Accessed: 2022-12-16, pp. 87–97. URL: http://nopr.niscpr.res.in/handle/123456789/9745.

[90] Weiyi Meng, Clement Yu, and King Lup Liu. "Building efficient and effective metasearch engines". In: *ACM Computing Surveys* Vol. 34, No. 1 (2002), pp. 48–89. DOI: 10.1145/505282.505284.

[91] Richard Peters and Robert Sikorski. "Metacrawler". In: *Science* Vol. 277, No. 5328 (1997), p. 977. DOI: 10.1109/64.577468.

[92] Tamar Sadeh. "Google Scholar versus metasearch systems". In: *High Energy Physics Libraries Webzine* Vol. 12, No. February (). Accessed: 2022-12-16, p. 2006. ISSN: 1424-2729. URL: https://webzine.web.cern.ch/12/.

[93] Milad Shokouhi and Luo Si. "Federated search". In: *Foundations and trends in information retrieval* Vol. 5, No. 1 (2011), pp. 1–102.

[94] Liwen Vaughan and Yue Chen. "Data mining from web search queries: A comparison of google trends and baidu index". In: *Journal of the Association for Information Science and Technology* Vol. 66, No. 1 (2015), pp. 13–22. DOI: 10.1002/asi.23201.

[95] Mahmudur Rahman. "Search Engines Going beyond Keyword Search: A Survey". In: *International Journal of Computer Applications* Vol. 75 (2013), pp. 1–8. DOI: 10.5120/13200-0357.

[96] Mercy Paul Selvan, Chandra Sekar, and Priya Dharshin. "Survey on Web Page Ranking Algorithms". In: *International Journal of Computer Applications* Vol. 41 (2012), pp. 1–7. DOI: 10.5120/5646-7764.

[97] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. *The PageRank citation ranking: Bringing order to the web.* Tech. rep. Accessed: 2022-12-16. Stanford InfoLab, 1999. URL: http://ilpubs.stanford.edu:8090/422/.

[98] Jon Kleinberg. "Authoritative sources in a hyperlinked environment." In: *Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*. Vol. 98. Accessed: 2022-12-16. Citeseer. 1998, pp. 668–677. URL: http://dl.acm.org/citation.cfm?id=314613.315045.

[99] Tony Abou Assaleh, Tapajyoti Das, Weizheng Gao, Yingbo Miao, Philip O'Brien, and Zhen Zhen. "A link-based ranking scheme for focused search". In: *Proceedings of the 16th International Conference on World Wide Web*. 2007, pp. 1125–1126. DOI: 10.1145/1242572.1242727.

[100] Burak Tokgoz, Zeynep Nur Ozcilnak, Cangla Cinar, Mustafa Tarcan Yalun, and Yıltan Bitirim. "An evaluation of Turkish retrieval performance of popular search engines for Internet and image search by using common lists". In: *Proceedings of the 3rd International Conference on Digital Information and Communication Technology and its Applications*. 2013, pp. 148–153. DOI: 10.1109/ICIW.2007.15.

[101] Aabid Hussain, Sumeer Gul, Tariq Ahmad Shah, and Sheikh Shueb. "Retrieval effectiveness of image search engines". In: *The Electronic Library* (2019). DOI: 10.1108/EL-07-2018-0142.

[102] Mehrdad CheshmehSohrabi and Elham Adnani Sadati. "Performance evaluation of web search engines in image retrieval: An experimental study". In: *Information Development* (2021). DOI: 10.1177/0266666921101021.

[103] Fuat Uluc, Erkan Emirzade, and Yiltan Bitirim. "The impact of number of query words on image search engines". In: *Proceedings of the 2nd International Conference on Internet and Web Appli-*

cations and Services (ICIW'07). IEEE. 2007, pp. 50–50. DOI: 10.1109/ICIW.2007.61.

[104] Ece Çakir, Huseyin Bahceci, and Yiltan Bitirim. "An evaluation of major image search engines on various query topics". In: *Proceedings of the 3rd International Conference on Internet Monitoring and Protection*. IEEE. 2008, pp. 161–165. DOI: 10.1109/ICIMP.2008.9.

[105] Arun Adrakatti, Ravindranath Wodeyar, and Kamalasab Mulla. "Search by image: a novel approach to content based image retrieval system". In: *International Journal of Library Science* Vol. 14, No. 3 (2016), pp. 41–47.

[106] Nirali Arora and Sharvari Govilkar. "Survey on different ranking algorithms along with their approaches". In: *International Journal of Computer Applications* Vol. 975 (2016), p. 8887. DOI: 10.5120/ijca2016908514.

[107] Muhammd Jawad Hamid Mughal. "Data Mining: Web Data Mining Techniques, Tools and Algorithms: An Overview". In: *International Journal of Advanced Computer Science and Applications* Vol. 9 (2018). DOI: 10.14569/IJACSA.2018.090630.

[108] Alan Ritter, Colin Cherry, and William Dolan. "Page Ranking Algorithms: A Survey". In: *Proceedings of the 2009 IEEE International Advance Computing Conference*. 2009, pp. 1530–1537. DOI: 10.1109/IADCC.2009.4809246.

[109] Tim Berners Lee, James Hendler, and Ora Lassila. "The semantic web". In: *Scientific American* Vol. 284, No. 5 (2001). Accessed: 2022-12-16, pp. 34–43. URL: http://www.krchowdhary.com/ai/ai16/sematic%5C%20web-sci-am.pdf.

[110] Ramanathan Guha, Rob Mc Cool, and Eric Miller. "Semantic search". In: *Proceedings of the 12th International Conference on World Wide Web*. 2003, pp. 700–709.

[111] Qazi Mudassar Ilyas, Yang Zong Kai, and Adeel Talib. "A conceptual architecture for semantic search engine". In: *Proceedings of the 8th International Multitopic Conference*. 2004, pp. 605–610. DOI: 10.1109/INMIC.2004.1492956.

[112] José Luis Sánchez Cervantes, Luis Omar Colombo Mendoza, Giner Alor Hernández, Jorge Luis Garcıa Alcaráz, José Marıa Álvarez-Rodrıguez, and Alejandro Rodrıguez-González. "LINDASearch: a faceted search system for linked open datasets". In: *Wireless Networks* Vol. 26, No. 8 (2020), pp. 5645–5663. DOI: 10.1007/s11276-019-02029-z.

[113] Sanjib Kumar Sahu, Durga Prasad Mahapatra, and Rakesh Chandra Balabantaray. "Comparative study of search engines in context of features and semantics". In: *Journal of Theoretical and Applied Information Technology* Vol. 88, No. 2 (2016). Accessed: 2022-12-16, pp. 210–218. URL: http://www.jatit.org/volumes/Vol88No2/3Vol88No2.pdf.

[114] Bzar Khidir Hussan. "Comparative study of semantic and keyword based search engines". In: *Advances in Science, Technology and Engineering Systems Journal* Vol. 5 (2020). Accessed: 2022-12-16, pp. 106–111. URL: https://www.astesj.com/publications/ASTESJ_050114.pdf.

[115] Ranjna Jain, Neelam Duhan, and Ajay Kumar Sharma. "Comparative study on semantic search engines". In: *International Journal of Computer Applications* Vol. 131, No. 14 (2015), pp. 4–11. DOI: 10.5120/ijca2015907370.

[116] Jeff Heflin and James Hendler. "Searching the Web with SHOE". In: *AAAI-2000 Workshop Proceedings on AI for Web Search*. 2000, pp. 35–40. DOI: 10.21236/ada440405.

[117] Elwin Huaman, Amar Tauqeer, and Anna Fensel. "Towards Knowledge Graphs Validation Through Weighted Knowledge Sources". In: *Proceedings of the Iberoamerican Knowledge Graphs and Semantic Web Conference*. Springer. 2021, pp. 47–60. DOI: 10.1007/978-3-030-91305-2_4.

[118] Subham Roy, Akshay Modak, Debabrata Barik, and Surajit Goon. "An overview of semantic search engines". In: *International Journal of Research and Review* Vol. 6, No. 10 (2019). Accessed: 2022-12-16, pp. 73–85. URL: https://www.ijrrjournal.com/IJRR_Vol.6_Issue.10_Oct2019/IJRR0012.pdf.

[119] Govathoti Sudeepthi, Guttula1 Anuradha, and Prasad Babu. "A survey on semantic web search engine". In: *International Journal of Computer Science Issues (IJCSI)* Vol. 9, No. 2 (2012), p. 241.

[120] Li Ding, Tim Finin, Anupam Joshi, Rong Pan, Scott Cost, Yun Peng, Pavan Reddivari, Vishal Doshi, and Joel Sachs. "Swoogle: a search and metadata engine for the semantic web". In: *Proceedings of the 13th ACM International Conference on Information and Knowledge Management*. 2004, pp. 652–659. DOI: 10.1145/1031171.1031289.

[121] Gong Cheng, Weiyi Ge, Honghan Wu, and Yuzhong Qu. "Searching Semantic Web objects based on class hierarchies". In: *Proceedings of the 2008 World Wide Web Workshop on Linked Data on the Web*. Vol. 369. Accessed: 2022-12-16. 2008. URL: https://ceur-ws.org/Vol-369/paper12.pdf.

[122] Mathieu Aquin, Li Ding, and Enrico Motta. "Semantic web search engines". In: *Handbook of Semantic Web Technologies*. 2011, pp. 659–700. DOI: 10.1007/978-3-540-92913-0_16.

[123] Kemafor Anyanwu, Angela Maduko, and Amit Sheth. "Semrank: ranking complex relationship search results on the semantic web". In: *Proceedings of the 14th International Conference on World Wide Web*. 2005, pp. 117–127. DOI: 10.1145/1060745.1060766.

[124] Brian McBride. "The resource description framework (RDF) and its vocabulary description language RDFS". In: *Handbook on ontologies*. Springer, 2004, pp. 51–65. DOI: 10.1007/978-3-540-24750-0_3.

[125] Dieter Fensel, Umutcan Simsek, Kevin Angele, Elwin Huaman, Elias Kärle, Oleksandra Panasiuk, Ioan Toma, Jürgen Umbrich, and Alexander Wahler. "Knowledge Graphs: Methodology, Tools and Selected Use Cases". In: *Knowledge Graphs* (2020). DOI: 10.1007/978-3-030-37439-6.

[126] Jeff Pan, Guido Vetere, Jose Manuel Gomez-Perez, and Honghan Wu. *Exploiting Linked Data and Knowledge Graphs in Large Organisations*. 2017. ISBN: 978-3-319-45652-2. DOI: 10.1007/978-3-319-45654-6.

[127] Sareh Aghaei, Kevin Angele, and Anna Fensel. "Finding Similar Entities across Knowledge Graphs". In: *Proceedings of the 7th International Conference on Advances in Computer Science and Information Technology*. AIRCC Publishing Corporation, 2021, pp. 01–11. DOI: 10.5121/csit.2021.110301.

[128] Elwin Huaman and Dieter Fensel. "Knowledge Graph Curation: A Practical Framework". In: *Proceedings of the 10th International*

Joint Conference on Knowledge Graphs. 2021, pp. 166–171. DOI: 10.1145/3502223.3502247.

[129] Siraya Sitthisarn, Lydia Lau, and Peter Dew. "A survey of semantic keyword search approaches". In: *International Journal of Applied Computer Technology and Information Systems* Vol. 2, No. 2 (2013), pp. 29–37.

[130] Fatemeh Lashkari, Faezeh Ensan, Ebrahim Bagheri, and Ali Ghorbani. "Efficient indexing for semantic search". In: *Expert Systems with Applications* Vol. 73 (2017), pp. 92–114. DOI: 10.1016/j.eswa.2016.12.033.

[131] Vikas Jindal, Seema Bawa, and Shalini Batra. "A review of ranking approaches for semantic search on Web". In: *Information Processing and Management* Vol. 50, No. 2 (2014), pp. 416–425. DOI: 10.1016/j.ipm.2013.10.004.

[132] Wang Wei, Payam Barnaghi, and Andrzej Bargiela. "Search with meanings: an overview of semantic search systems". In: *International journal of Communications* Vol. 3 (2008). Accessed: 2022-12-16, pp. 76–82. URL: http://www.bargiela.com/papers/a35.pdf.

[133] José Wellington Franco Silva, Amanda Drielly Pires Venceslau, Juliano Efson Sales, José Gilvan Rodrigues Maia, Vládia Célia Monteiro Pinheiro, and Vânia Maria Ponte Vidal. "A short survey on end-to-end simple question answering systems". In: *Artificial Intelligence Review* (2020), pp. 1–25. DOI: 10.1007/s10462-020-09826-5.

[134] Konrad Höffner, Sebastian Walter, Edgard Marx, Ricardo Usbeck, Jens Lehmann, and Axel Cyrille Ngonga Ngomo. "Survey on chal-

lenges of question answering in the semantic web". In: *Semantic Web* Vol. 8, No. 6 (2017), pp. 895–920. DOI: 10.3233/SW-160247.

[135] Eleftherios Dimitrakis, Konstantinos Sgontzos, and Yannis Tzitzikas. "A survey on question answering systems over linked data and documents". In: *Journal of Intelligent Information Systems* Vol. 55 (2019), pp. 233–259. DOI: 10.1007/s10844-019-00584-7.

[136] Amit Mishra and Sanjay Kumar Jain. "A survey on question answering systems with classification". In: *Journal of King Saud University - Computer and Information* Vol. 28 (2016), pp. 345–361. DOI: 10.1016/j.jksuci.2014.10.007.

[137] Álvaro Rodrigo and Anselmo Peñas. "A study about the future evaluation of Question-Answering systems". In: *Knowledge-Based Systems* Vol. 137 (2017), pp. 83–93. DOI: 10.1016/j.knosys.2017.09.015.

[138] Zahra Abbasiyantaeb and Saeedeh Momtazi. "Text-based Question Answering from Information Retrieval and Deep Neural Network Perspectives: A Survey". In: *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* Vol. 11 (2020). DOI: 10.1002/widm.1412.

[139] Marcelo Arenas, Bernardo Cuenca Grau, Evgeny Kharlamov, Sarunas Marciuska, and Dmitriy Zheleznyakov. "Faceted search over RDF-based knowledge graphs". In: *Journal of Web Semantic* (2016), pp. 55–74. DOI: 10.1016/j.websem.2015.12.002.

[140] Fabian Suchanek, Gjergji Kasneci, and Gerhard Weikum. "Yago: A core of semantic knowledge unifying wordnet and wikipedia". In: *Proceedings of the 16th International World Wide Web Conference*. 2007, pp. 697–706. DOI: 10.1145/1242572.1242667.

[141] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. "Freebase: a collaboratively created graph database for structuring human knowledge". In: *Proceedings of the 2008 ACM SIGMOD International Conference on Management of data*. 2008, pp. 1247–1250. DOI: 10.1145/1376616.1376746.

[142] Lisa Ehrlinger and Wolfram Wöß. "Towards a Definition of Knowledge Graphs." In: *Proceedings of the SEMANTiCS (Posters, Demos, SuCCESS)*. Vol. 48. 1-4. Accessed: 2022-12-16. Citeseer, 2016, p. 2. URL: https://ceur-ws.org/Vol-1695/paper4.pdf.

[143] Tahseen Sultana and Srinivasu Badugu. "A review on different question answering system approaches". In: *Advances in Decision Sciences, Image Processing, Security and Computer Vision*. Springer, 2020, pp. 579–586. DOI: 10.1007/978-3-030-24318-0_67.

[144] Tom Kenter, Alexey Borisov, Christophe Van Gysel, Mostafa Dehghani, Maarten de Rijke, and Bhaskar Mitra. "Neural networks for information retrieval". In: *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2017, pp. 1403–1406. DOI: 10.48550/arXiv.1801.02178.

[145] Li Dong and Mirella Lapata. "Language to logical form with neural attention". In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*. 2016, pp. 33–43. DOI: 10.18653/v1/P16-1004.

[146] Chandra Obula Reddy and Madhavi Latha. "A Survey on types of question answering system". In: *IOSR Journal of Computer Engineering* Vol. 19, No. 6 (2017), pp. 19–23. DOI: 10.48550/arXiv.1911.04879.

[147] Ammar Arbaaeen and Asadullah Shah. "Ontology-Based Approach to Semantically Enhanced Question Answering for Closed Domain: A Review". In: *Information* Vol. 12, No. 5 (2021), p. 200. DOI: 10.3390/info12050200.

[148] Michael Woodley, Jante Nijenhuis, and Raegan Murphy. "The Victorians were still faster than us. Commentary: Factors influencing the latency of simple reaction time". In: *Frontiers in Human Neuroscience* Vol. 9 (2015). DOI: 10.3389/fnhum.2015.00452.

[149] Marek Hrúz, Pavel Campr, Zdenek Krňoul, Milos Železnỳ, Oya Aran, and Pinar Santemiz. "Multi-modal dialogue system with sign language capabilities". In: *Proceedings of the 13th International ACM SIGACCESS Conference on Computers and Accessibility*. 2011, pp. 265–266. DOI: 10.1145/2049536.2049599.

[150] Gabriel Skantze and David Schlangen. "Incremental dialogue processing in a micro-domain". In: *Proceedings of the 12th Conference of the European Chapter of the ACL*. 2009, pp. 745–753. DOI: 10.3115/1609067.1609150.

[151] Angus Addlesee, Arash Eshghi, and Ioannis Konstas. "Current challenges in spoken dialogue systems and why they are critical for those living with dementia". In: *ArXiv* (2019). DOI: 10.48550/arXiv.1909.06644.

[152] Jan Deriu, Álvaro Rodrigo, Arantxa Otegi, Guillermo Echegoyen, Rosset, Eneko Agirre, and Mark Cieliebak. "Survey on evaluation methods for dialogue systems". In: *Artificial Intelligence Review* Vol. 54 (2020), pp. 755–810. DOI: 10.1007/s10462-020-09866-x.

[153] Li Zhou, Jianfeng Gao, Di Li, and Heung Yeung Shum. "The design and implementation of xiaoice, an empathetic social chat-

bot". In: *Computational Linguistics* Vol. 46, No. 1 (2020), pp. 53–93. DOI: 0.1162/COLI_a_00368.

[154] Stavros Mallios and Nikolaos Bourbakis. "A survey on human machine dialogue systems". In: *Proceedings of the 7th International Conference on Information, Intelligence, Systems & Applications*. 2016, pp. 1–7. DOI: 10.1109/IISA.2016.7785371.

[155] Ana Paula Chaves and Marco Aurelio Gerosa. "The Impact of Chatbot Linguistic Register on User Perceptions: A Replication Study". In: *International Workshop Proceedings on Chatbot Research and Design*. Springer. 2021, pp. 143–159. DOI: 10.1007/978-3-030-94890-0_9.

[156] Christine Liebrecht, Lena Sander, and Charlotte van Hooijdonk. "Too informal? How a chatbot's communication style affects brand attitude and quality of interaction". In: *International Workshop Proceedings on Chatbot Research and Design*. Springer. 2020, pp. 16–31. DOI: 10.1007/978-3-030-68288-0_2.

[157] Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. "Convolutional Neural Network Architectures for Matching Natural Language Sentences". In: *Proceedings of the 27th International Conference on Neural Information Processing Systems*. NIPS'14. Montreal, Canada: MIT Press, 2014, pp. 2042–2050. DOI: 10.5555/2969033.2969055.

[158] Ryan Lowe, Nissan Pow, Iulian Serban, and Joelle Pineau. "The Ubuntu Dialogue Corpus: A Large Dataset for Research in Unstructured Multi-Turn Dialogue Systems". In: *Proceedings of the 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*. 2015, pp. 285–294. DOI: 10.18653/v1/W15-4640. URL: https://aclanthology.org/W15-4640.

[159] Alan Ritter, Colin Cherry, and William Dolan. "Data-driven response generation in social media". In: *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*. Accessed: 2022-12-16. 2011, pp. 583–593. URL: https://aclanthology.org/D11-1054.

[160] Yiping Song, Rui Yan, Xiang Li, Dongyan Zhao, and Ming Zhang. "Two are Better than One: An Ensemble of Retrieval- and Generation-Based Dialog Systems". In: *ArXiv* Vol. abs/1610.07149 (2016). DOI: 10.48550/arXiv.1610.07149.

[161] Minghui Qiu, Feng-Lin Li, Siyu Wang, Xing Gao, Yan Chen, Weipeng Zhao, Haiqing Chen, Jun Huang, and Wei Chu. "Alime chat: A sequence to sequence and rerank based chatbot engine". In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*. 2017, pp. 498–503. DOI: 10.18653/v1/P17-2079.

[162] Jinjie Ni, Tom Young, Vlad Pandelea, Fuzhao Xue, Vinay Adiga, and Erik Cambria. "Recent advances in deep learning based dialogue systems: A systematic survey". In: *ArXiv* (2021). DOI: 10.48550/arXiv.2105.0438.

[163] Samuel Coope, Tyler Farghly, Daniela Gerz, Ivan Vulić, and Matthew Henderson. "Span-ConveRT: Few-shot Span Extraction for Dialog with Pretrained Conversational Representations". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 2020, pp. 107–121. DOI: 10.18653/v1/2020.acl-main.11.

[164] Lisong Qiu, Juntao Li, Wei Bi, Dongyan Zhao, and Rui Yan. "Are training samples correlated? learning to generate dialogue

responses with multiple references". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 2019, pp. 3826–3835. DOI: 10.18653/v1/P19-1372.

[165] Seungwhan Moon, Pararth Shah, Anuj Kumar, and Rajen Subba. "Opendialkg: Explainable conversational reasoning with attention-based walks over knowledge graphs". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 2019, pp. 845–854.

[166] Hainan Zhang, Yanyan Lan, Liang Pang, Jiafeng Guo, and Xueqi Cheng. "Recosa: Detecting the relevant contexts with self-attention for multi-turn dialogue generation". In: *ArXiv* (2019). DOI: 10.48550/arXiv.1907.05339.

[167] Matthew Henderson, Ivan Vulić, Daniela Gerz, Iñigo Casanueva, Paweł Budzianowski, Sam Coope, Georgios Spithourakis, Tsung Hsien Wen, Nikola Mrkšić, and Pei Hao Su. "Training Neural Response Selection for Task-Oriented Dialogue Systems". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 2019, pp. 5392–5404. DOI: 10.18653/v1/P19-1536.

[168] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei Rusu, Joel Veness, Marc Bellemare, Alex Graves, Martin Riedmiller, Andreas Fidjeland, and Georg Ostrovski. "Human-level control through deep reinforcement learning". In: *nature* Vol. 518, No. 7540 (2015), pp. 529–533. DOI: 10.1038/nature14236.

[169] Jaehun Jung, Bokyung Son, and Sungwon Lyu. "Attnio: Knowledge graph exploration with in-and-out attention flow for knowledge-grounded dialogue". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*

(EMNLP). 2020, pp. 3484–3497. DOI: 10.18653/v1/2020.emnlp-main.280.

[170] Joseph Weizenbaum. "ELIZA — a computer program for the study of natural language communication between man and machine". In: *Communications of the ACM* Vol. 9, No. 1 (1983), pp. 36–45. DOI: 10.1145/365153.365168.

[171] Tomá Zemík. "A Brief History of Chatbots". In: *DEStech Transactions on Computer Science and Engineering* (2019).

[172] Guendalina Caldarini, Sardar Jaf, and Kenneth McGarry. "A Literature Survey of Recent Advances in Chatbots". In: *Information* Vol. 13, No. 1 (2022), p. 41. DOI: 10.3390/info13010041.

[173] Ketakee Nimavat and Tushar Champaneria. "Chatbots: An overview. Types, architecture, tools and future possibilities". In: *International Journal for Scientific Research & Development* Vol. 5, No. 7 (2017). Accessed: 2022-12-16, pp. 1019–1024. URL: https://www.ijsrd.com/articles/IJSRDV5I70501.pdf.

[174] Adam Miner, Liliana Laranjo, and Baki Kocaballi. "Chatbots in the fight against the COVID-19 pandemic". In: *Nature Partner Journals series (NPJ) Digital Medicine* Vol. 3 (2020). DOI: 10.1038/s41746-020-0280-0.

[175] Eleni Adamopoulou and Lefteris Moussiades. "An Overview of Chatbot Technology". In: *Artificial Intelligence Applications and Innovations* Vol. 584 (2020), pp. 373–383. DOI: 10.1007/978-3-030-49186-4_31.

[176] Aditya Deshpande, Alisha Shahane, Darshana Gadre, Mrunmayi Deshpande, and Prachi Joshi. "A survey of various chatbot implementation techniques". In: *International Journal of Computer Engineering and Applications* Vol. 11, No. 7 (2017). Accessed:

2022-12-16. ISSN: 321-3469. URL: https://www.ijcea.com/wp-content/uploads/2017/07/ID_66_Draft_1_final.pdf.

[177] Prissadang Suta, Xi Lan, Biting Wu, Pornchai Mongkolnam, and Jh Chan. "An overview of machine learning in chatbots". In: *International Journal of Mechanical Engineering and Robotics Research* Vol. 9, No. 4 (2020), pp. 502–510. DOI: 10.18178/ijmerr.9.4.502-510.

[178] Jiyeon Kim and Sandra Forsythe. "Adoption of virtual try-on technology for online apparel shopping". In: *Journal of Interactive Marketing* Vol. 22, No. 2 (2008), pp. 45–59. DOI: 10.1002/dir.20113.

[179] Hyun Joo Lee and Kiseol Yang. "Interpersonal service quality, self-service technology (SST) service quality, and retail patronage". In: *Journal of Retailing and Consumer Services* Vol. 20, No. 1 (2013), pp. 51–57. DOI: 10.1016/j.jretconser.2012.10.005.

[180] Kiseol Yang and Judith Forney. "The moderating role of consumer technology anxiety in mobile shopping adoption: differential effects of facilitating conditions and social influences". In: *Journal of Electronic Commerce Research* Vol. 14, No. 4 (2013), p. 334.

[181] Wynne Chin, Barbara Marcolin, and Peter Newsted. "A partial least squares latent variable modeling approach for measuring interaction effects: Results from a Monte Carlo simulation study and an electronic-mail emotion/adoption study". In: *Information systems research* Vol. 14, No. 2 (2003), pp. 189–217. DOI: 10.1287/isre.14.2.189.16018.

[182] Lin Li, Kyung Young Lee, Emmanuel Emokpae, and Sung-Byung Yang. "What makes you continuously use chatbot services? Evidence from chinese online travel agencies". In: *Electronic Markets*

Vol. 31, No. 3 (2021), pp. 575–599. DOI: 10.1007/s12525-020-00454-z.

[183] Mohit Jain, Pratyush Kumar, Ramachandra Kota, and Shwetak Naran Patel. "Evaluating and informing the design of chatbots". In: *Proceedings of 2018 Designing Interactive Systems Conference*. 2018, pp. 895–906. DOI: 10.1145/3196709.3196735.

[184] Reshmi Sankar and Kannan Balakrishnan. "Empowering chatbots with business intelligence by big data integration". In: *International Journal of Advanced Research in Computer Science* Vol. 9 (2018), pp. 627–631. DOI: 10.26483/ijarcs.v9i1.5398.

[185] Arno Breitfuss, Karen Errou, Anelia Kurteva, and Anna Fensel. "Representing emotions with knowledge graphs for movie recommendations". In: *Future Generation Computer Systems* Vol. 125 (2021), pp. 715–725. ISSN: 0167-739X. DOI: 10.1016/j.future.2021.06.001.

[186] Addi Ait Mlouk and Lili Jiang. "KBot: A Knowledge Graph Based ChatBot for Natural Language Understanding Over Linked Data". In: *IEEE Access* (2020), pp. 149220–149230. DOI: 10.1109/ACCESS.2020.3016142.

[187] Rui Yan and Wei Wu. "Empowering conversational AI is a trip to Mars: Progress and future of open domain human-computer dialogues". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 35. 17. 2021, pp. 15078–15086. DOI: 10.1609/aaai.v35i17.17771.

[188] *The Rise of Multimodal and Multilingual Search*. Accessed: 2022-01-23. URL: https://www.oncrawl.com/technical-seo/rise-multimodal-multilingual-search/.

[189] *ReadWrite What Does the Future of Online Search Look Like?* Accessed: 2021-07-30. URL: https://readwrite.com/2020/12/28/future-of-seo/.

[190] *The Future of Search*. Accessed: 2021-07-30. URL: https://www.iprospect.com/en/gb/news-and-insights/news/the-future-of-search/.

[191] Mark Sanderson and Bruce Croft. "The history of information retrieval research". In: *Proceedings of the IEEE* Vol. 100, No. Special Centennial Issue (2012), pp. 1444–1451. DOI: 10.1109/JPROC.2012.2189916.

[192] Martí Cuquet and Anna Fensel. "The societal impact of big data: A research roadmap for Europe". In: *Technology in Society* Vol. 54 (2018), pp. 74–86. ISSN: 0160-791X. DOI: 10.1016/j.techsoc.2018.03.005.

[193] Patrick Reid and Des Laffey. "Search Engines: Past, Present, and Future". In: *Encyclopedia of E-Commerce Development, Implementation, and Management*. 2016, pp. 1102–1115. DOI: 0.4018/978-1-61520-611-7.ch126.

[194] Richard Jacques, Asbjørn Følstad, Elizabeth Gerber, Jonathan Grudin, Ewa Luger, Andrés Monroy Hernández, and Dakuo Wang. "Conversational agents: Acting on the wave of research and development". In: *Proceedings of the 2019 China Conference on Human Factors in Computing Systems*. 2019, pp. 1–8. DOI: 10.1145/3290607.3299034.

[195] Julia Masche and Nguyen Thinh Le. "A review of technologies for conversational systems". In: *Proceedings of the International Conference on Computer Science, Applied Mathematics and Ap-*

plications. Springer. 2017, pp. 212–225. DOI: 10.1007/978-3-319-61911-8_19.

[196] Shaoxiong Ji, Shirui Pan, Erik Cambria, Pekka Marttinen, and Yu Philip. "A survey on knowledge graphs: Representation, acquisition, and applications". In: *IEEE Transactions on Neural Networks and Learning Systems* (2021). DOI: 10.1109/TNNLS.2021.3070843.

[197] Bilal Abu Salih. "Domain-specific knowledge graphs: A survey". In: *Journal of Network and Computer Applications* Vol. 185 (2021), p. 103076. DOI: 10.1016/j.jnca.2021.103076.

[198] Graham Klyne. "Resource description framework (RDF): Concepts and abstract syntax". In: *http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/* (2004).

[199] Hiba Arnaout and Shady Elbassuoni. "Effective searching of RDF knowledge graphs". In: *Journal of Web Semantics* Vol. 48 (2018), pp. 66–84. DOI: 10.1016/j.websem.2017.12.001.

[200] Martin Mansfield, Valentina Tamma, Phil Goddard, and Frans Coenen. "Capturing Expert Knowledge for Building Enterprise SME Knowledge Graphs". In: *Proceedings of the 11th on Knowledge Capture Conference*. 2021, pp. 129–136. DOI: 10.1145/3460210.3493569.

[201] Anna Fensel, Zaenal Akbar, Elias Kärle, Christoph Blank, Patrick Pixner, and Andreas Gruber. "Knowledge graphs for online marketing and sales of touristic services". In: *Information* Vol. 11, No. 5 (2020), p. 253. DOI: 10.3390/info11050253.

[202] Dinghe Xiao, Nannan Wang, Jiangang Yu, Chunhong Zhang, and Jiaqi Wu. "A Practice of Tourism Knowledge Graph Construction Based on Heterogeneous Information". In: *Proceedings of the*

China National Conference on Chinese Computational Linguistics. Springer. 2020, pp. 159–173. DOI: 10.1007/978-3-030-63031-7_12.

[203] Chun Lu, Philippe Laublet, and Milan Stankovic. "Travel attractions recommendation with knowledge graphs". In: *European Knowledge Acquisition Workshop Proceedings*. Springer. 2016, pp. 416–431. DOI: doi.org/10.1007/978-3-319-49004-5_27.

[204] Xuchao Liang, Han Cao, and Weizhen Zhang. "Knowledge extraction experiment based on tourism knowledge graph Q & A data set". In: *Proceedings of the 2020 IEEE International Conference on Power, Intelligent Computing and Systems (ICPICS)*. IEEE. 2020, pp. 828–832. DOI: 10.1109/ICPICS50287.2020.9202197.

[205] Elias Kärle, Umutcan Şimşek, Oleksandra Panasiuk, and Dieter Fensel. "Building an ecosystem for the tyrolean tourism knowledge graph". In: *Proceedings of the International Conference on Web Engineering*. Springer. 2018, pp. 260–267. DOI: 10.1007/978-3-030-03056-8_25.

[206] Weitao Zhang, Tianlong Gu, Wenping Sun, Yochum Phatpicha, Liang Chang, and Chenzhong Bin. "Travel attractions recommendation with travel spatial-temporal knowledge graphs". In: *Proceedings of the International Conference of pioneering computer scientists, engineers and educators*. Springer. 2018, pp. 213–226. DOI: 10.1007/978-981-13-2206-8_19.

[207] Sen Hu, Lei Zou, Jeffrey Xu Yu, Haixun Wang, and Dongyan Zhao. "Answering natural language questions by subgraph matching over knowledge graphs". In: *IEEE Transactions on Knowledge and Data Engineering* Vol. 30, No. 5 (2017), pp. 824–837. DOI: 10.1109/TKDE.2017.2766634.

[208] Addi Ait Mlouk and Lili Jiang. "KBot: a Knowledge graph based chatBot for natural language understanding over linked data". In: *IEEE Access* Vol. 8 (2020), pp. 149220–149230. DOI: 10.1109/ACCESS.2020.3016142.

[209] Bolanle Ojokoh and Emmanuel Adebisi. "A review of question answering systems". In: *Journal of Web Engineering* Vol. 17, No. 8 (2018), pp. 717–758. DOI: 10.13052/jwe1540-9589.1785.

[210] Yunqi Qiu, Yuanzhuo Wang, Xiaolong Jin, and Kun Zhang. "Stepwise reasoning for multi-relation question answering over knowledge graph with weak supervision". In: *Proceedings of the 13th International Conference on Web Search and Data Mining*. 2020, pp. 474–482. DOI: 10.1145/3336191.3371812.

[211] Christina Unger, Lorenz Bühmann, Jens Lehmann, Axel Cyrille Ngonga Ngomo, Daniel Gerber, and Philipp Cimiano. "Template-based question answering over RDF data". In: *Proceedings of the 21st International Conference on World Wide Web*. 2012, pp. 639–648. DOI: 10.1145/2187836.2187923.

[212] Weiguo Zheng, Jeffrey Xu Yu, Lei Zou, and Hong Cheng. "Question answering over knowledge graphs: question understanding via template decomposition". In: *Proceedings of the VLDB Endowment* Vol. 11, No. 11 (2018), pp. 1373–1386. DOI: 10.14778/3236187.3236192.

[213] Wanyun Cui, Yanghua Xiao, Haixun Wang, Yangqiu Song, Seung-won Hwang, and Wei Wang. "KBQA: Learning Question Answering over QA Corpora and Knowledge Bases". In: Vol. 10, No. 5 (2017), pp. 565–576. ISSN: 2150-8097. DOI: 10.14778/3055540.3055549.

[214] Shiyan Ou, Constantin Orasan, Dalila Mekhaldi, and Laura Hasler. "Automatic Question Pattern Generation for Ontology-based Question Answering." In: *Proceedings of the Flairs Conference*. 2008, pp. 183–188.

[215] Hannah Bast and Elmar Haussmann. "More accurate question answering on freebase". In: *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. 2015, pp. 1431–1440. DOI: 10.1145/2806416.2806472.

[216] Michihiro Yasunaga, Hongyu Ren, Antoine Bosselut, Percy Liang, and Jure Leskovec. "QA-GNN: Reasoning with Language Models and Knowledge Graphs for Question Answering". In: *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, 2021, pp. 535–546. DOI: 10.18653/v1/2021.naacl-main.45.

[217] Ram Ganesan Athreya, Srividya Bansal, Axel Cyrille Ngomo Ngonga, and Ricardo Usbeck. "Template-based question answering using recursive neural networks". In: *Proceedings of the 2021 IEEE 15th International Conference on Semantic Computing*. IEEE. 2021, pp. 195–198. DOI: 10.1109/ICSC50631.2021.00041.

[218] Priyansh Trivedi, Gaurav Maheshwari, Mohnish Dubey, and Jens Lehmann. "Lc-quad: A corpus for complex question answering over knowledge graphs". In: *Proceedings of the International Semantic Web Conference*. Springer. 2017, pp. 210–218. DOI: 10.1007/978-3-319-68204-4_22.

[219] Akshay Kumar Dileep, Anurag Mishra, Ria Mehta, Siddharth Uppal, Jaydeep Chakraborty, and Srividya K Bansal. "Template-based Question Answering analysis on the LC-QuAD2. 0

Dataset". In: *Proceedings of the 2021 IEEE 15th International Conference on Semantic Computing (ICSC)*. IEEE. 2021, pp. 443–448. DOI: 10.1109/ICSC50631.2021.00079.

[220] 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: *Proceedings of the International Semantic Web Conference*. Springer. 2019, pp. 69–78. DOI: 10.1007/978-3-030-30796-7_5.

[221] Leo Breiman. "Random forests". In: *Machine learning* Vol. 45, No. 1 (2001), pp. 5–32. DOI: 10.1023/A:1010933404324.

[222] Daniel Vollmers, Rricha Jalota, Diego Moussallem, Hardik Topiwala, Axel Cyrille Ngonga Ngomo, and Ricardo Usbeck. "Knowledge Graph Question Answering using Graph-Pattern Isomorphism". In: *Proceedings of the 17th International Conference on Semantic Systems*. Vol. 53. 2021, p. 103. DOI: 10.3233/SSW210038.

[223] Pablo Mendes, Max Jakob, Andrés Garcıa Silva, and Christian Bizer. "DBpedia spotlight: shedding light on the web of documents". In: *Proceedings of the 7th International Conference on Semantic Systems*. 2011, pp. 1–8. DOI: 10.1145/2063518.2063519.

[224] Tianqi Chen and Carlos Guestrin. "Xgboost: A scalable tree boosting system". In: *Proceedings of the 22nd ACM Special Interest Group on Knowledge Discovery and Data Mining (SIGKDD) International Conference on Knowledge Discovery and Data Mining*. 2016, pp. 785–794. DOI: 10.1145/2939672.2939785.

[225] Hanning Gao, Lingfei Wu, Po Hu, and Fangli Xu. "RDF-to-text generation with graph-augmented structural neural encoders". In:

Proceedings of the 29th International Conference on International Joint Conferences on Artificial Intelligence. 2021, pp. 3030–3036. DOI: 10.24963/ijcai.2020/419.

[226] Yu Chen, Lingfei Wu, and Mohammed Zaki. "Reinforcement learning based graph-to-sequence model for natural question generation". In: *ArXiv* (2019). DOI: 10.48550/arXiv.1910.08832.

[227] Kun Xu, Lingfei Wu, Zhiguo Wang, Yansong Feng, Michael Witbrock, and Vadim Sheinin. "Graph2seq: Graph to sequence learning with attention-based neural networks". In: *ArXiv* (2018). DOI: 10.48550/arXiv.1804.00823.

[228] Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. "Semantic parsing on freebase from question-answer pairs". In: *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Accessed: 2022-12-16. 2013, pp. 1533–1544. URL: https://aclanthology.org/D13-1160.

[229] Junwei Bao, Nan Duan, Zhao Yan, Ming Zhou, and Tiejun Zhao. "Constraint-based question answering with knowledge graph". In: *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. Accessed: 2022-12-16. 2016, pp. 2503–2514. URL: https://aclanthology.org/C16-1236.

[230] Wen Tau Yih, Matthew Richardson, Christopher Meek, Ming Wei Chang, and Jina Suh. "The value of semantic parse labeling for knowledge base question answering". In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*. 2016, pp. 201–206. DOI: 10.18653/v1/P16-2033.

[231] Alon Talmor and Jonathan Berant. "The Web as a Knowledge-Base for Answering Complex Questions". In: *Proceedings of the*

2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. Association for Computational Linguistics, 2018, pp. 641–651. DOI: 10.18653/v1/N18-1059.

[232] Vladimir Levenshtein. "Binary codes capable of correcting deletions, insertions, and reversals". In: *Soviet physics doklady*. Vol. 10. 8. Soviet Union. 1966, pp. 707–710.

[233] Li Yujian and Liu Bo. "A normalized Levenshtein distance metric". In: *IEEE transactions on pattern analysis and machine intelligence* Vol. 29, No. 6 (2007), pp. 1091–1095. DOI: 10.1109/TPAMI.2007.1078.

[234] Nils Reimers and Iryna Gurevych. "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Association for Computational Linguistics, 2019, pp. 3982–3992. DOI: 10.18653/v1/D19-1410.

[235] Junwei Bao, Nan Duan, Ming Zhou, and Tiejun Zhao. "Knowledge-based question answering as machine translation". In: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*. 2014, pp. 967–976. DOI: 10.3115/v1/P14-1091.

[236] Anna Fensel, Ioan Toma, José Maria Garcia, Ioannis Stavrakantonakis, and Dieter Fensel. "Enabling customers engagement and collaboration for small and medium-sized enterprises in ubiquitous multi-channel ecosystems". In: *Computers in Industry* Vol. 65, No. 5 (2014), pp. 891–904. DOI: 10.1016/j.compind.2014.02.001.

[237] Apoorv Saxena, Aditay Tripathi, and Partha Talukdar. "Improving multi-hop question answering over knowledge graphs using knowledge base embeddings". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 2020, pp. 4498–4507. DOI: 10.18653/v1/2020.acl-main.412.

[238] Mahdi Bakhshi, Mohammadali Nematbakhsh, Mehran Mohsenzadeh, and Amir Masoud Rahmani. "Data-driven construction of SPARQL queries by approximate question graph alignment in question answering over knowledge graphs". In: *Expert Systems with Applications* Vol. 146 (2020), p. 113205. DOI: 10.1016/j.eswa.2020.113205.

[239] Kun Fu, Tingyun Mao, Yang Wang, Daoyu Lin, Yuanben Zhang, Junjian Zhan, Xian Sun, and Feng Li. "TS-Extractor: large graph exploration via subgraph extraction based on topological and semantic information". In: *Journal of Visualization* Vol. 24, No. 1 (2021), pp. 173–190. DOI: 10.1007/s12650-020-00699-y.

[240] Reid Andersen, Fan Chung, and Kevin Lang. "Using pagerank to locally partition a graph". In: *Internet Mathematics* Vol. 4, No. 1 (2007), pp. 35–64. DOI: 10.1080/15427951.2007.10129139.

[241] Jing Li, Aixin Sun, Jianglei Han, and Chenliang Li. "A survey on deep learning for named entity recognition". In: *IEEE Transactions on Knowledge and Data Engineering* Vol. 34 (2018), pp. 50–70. DOI: 10.1109/TKDE.2020.2981314.

[242] Xiao Ling, Sameer Singh, and Daniel Weld. "Design challenges for entity linking". In: *Transactions of the Association for Computational Linguistics* Vol. 3 (2015), pp. 315–328. DOI: 10.1162/tacl_a_00141.

[243] Alexander Miller, Adam Fisch, Jesse Dodge, Amir Hossein Karimi, Antoine Bordes, and Jason Weston. "Key-Value Memory Networks for Directly Reading Documents". In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2016, pp. 1400–1409. DOI: 10.18653/v1/D16-1147.

[244] Christina Unger, Axel Cyrille Ngonga Ngomo, and Elena Cabrio. "6th open challenge on question answering over linked data (qald-6)". In: *Proceedings of the 2016 Semantic Web Evaluation Challenge*. Springer. 2016, pp. 171–177. DOI: 10.1007/978-3-319-46565-4_13.

[245] Liang Shiqi, Kurt Stockinger, Maria Anisimova, and Manuel Gil. "Querying knowledge graphs in natural language". In: *Journal of Big Data* Vol. 8, No. 1 (2021). DOI: 10.1186/s40537-020-00383-w.

[246] Kevin Angele, Manuel Meitinger, Marc Bußjäger, Stephan Föhl, and Anna Fensel. "GraphSPARQL: a GraphQL interface for linked data". In: *Proceedings of the 37th ACM/SIGAPP Symposium on Applied Computing*. 2022, pp. 778–785. DOI: 10.1145/3477314.3507655.

[247] Sareh Aghaei. "Question Answering over Knowledge Graphs". In: *Proceedings of the 5th Doctoral Consortium at Rules and Reasoning (RuleML+RR)* Vol. 2956 (2021). Accessed: 2022-12-16. URL: http://ceur-ws.org/Vol-2956/paper35.pdf.

[248] Yunshi Lan, Gaole He, Jinhao Jiang, Jing Jiang, Wayne Xin Zhao, and Jirong Wen. "A Survey on Complex Knowledge Base Question Answering: Methods, Challenges and Solutions". In: *Proceedings*

of the International Joint Conference on Artificial Intelligence. 2021. DOI: DOI:10.24963/ijcai.2021/611.

[249] Chen Liang, Jonathan Berant, Quoc Le, Kenneth Forbus, and Ni Lao. "Neural Symbolic Machines: Learning Semantic Parsers on Freebase with Weak Supervision". In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 2017, pp. 23–33. DOI: 10.18653/v1/P17-1003.

[250] Romain Beaumont, Brigitte Grau, and Anne Laure Ligozat. "SemGraphQA@ QALD5: LIMSI participation at QALD5@ CLEF." In: *Proceedings of the 6th Conference and Labs of the Evaluation Forum*. Accessed: 2022-12-16. 2015. URL: https://hal.science/hal-02289245.

[251] Daniel Sussman, Youngser Park, Carey Priebe, and Vince Lyzinski. "Matched filters for noisy induced subgraph detection". In: *IEEE transactions on pattern analysis and machine intelligence* Vol. 42, No. 11 (2019), pp. 2887–2900. DOI: 10.1109/TPAMI.2019.2914651.

[252] Kaspar Riesen, Xiaoyi Jiang, and Horst Bunke. "Exact and inexact graph matching: Methodology and applications". In: *Managing and Mining Graph Data* (2010), pp. 217–247. DOI: 10.1007/978-1-4419-6045-0_7.

[253] Roberto Poli, Michael Healy, and Achilles Kameas. "Theory and applications of ontology: Computer applications". In: Springer, 2010, pp. 231–243. ISBN: 9048188466.

[254] Giuseppe Mazzeo and Carlo Zaniolo. "Answering Controlled Natural Language Questions on RDF Knowledge Bases." In: *Pro-*

ceedings of the International Conference on Extending Database Technology. 2016, pp. 608–611. DOI: 10.5441/002/edbt.2016.60.

[255] Hauke Hanke and Dorothee Knees. "A phase-field damage model based on evolving microstructure". In: *Asymptotic Analysis* Vol. 101 (2017), pp. 149–180. DOI: 10.3233/ASY-161396.

[256] Lefever Els. "A hybrid approach to domain-independent taxonomy learning". In: *Applied Ontology* Vol. 11, No. 3 (2016), pp. 255–278. ISSN: 1570-5838. DOI: 10.3233/AO-160170.

[257] Santha Sheela and Jaya Kumar. "Comparative Study of Syntactic Search Engine and Semantic Search Engine: A Survey". In: *Proceedings of the 2019 Fifth International Conference on Science Technology Engineering and Mathematics*. Vol. 1. 2019, pp. 1–4. DOI: 10.1109/ICONSTEM.2019.8918837.

[258] Daniel Jurafsky. "Speech & language processing". In: *Pearson Education India* (2000). Accessed: 2022-12-16. URL: https://web.stanford.edu/~jurafsky/slp3/ed3book.pdf.

[259] Li Zhou, Jianfeng Gao, Di Li, and Harry Shum. "The Design and Implementation of XiaoIce, an Empathetic Social Chatbot". In: *Computational Linguistics* Vol. 46(1) (2020), pp. 53–93. DOI: 10.1162/COLI_a_00368.

[260] Amit Mishra and Sanjay Kumar Jain. "A survey on question answering systems with classification". In: *Journal of King Saud University-Computer and Information Sciences* Vol. 28, No. 3 (2016), pp. 345–361. DOI: 10.1016/j.jksuci.2014.10.007.

[261] Michael Good, John Whiteside, Dennis Wixon, and Sandra Jones. "Building a user-derived interface". In: *Communications of the ACM* Vol. 27, No. 10 (1984), pp. 1032–1043. DOI: 10.1145/358274.358284.

[262] John Gould, John Conti, and Todd Hovanyecz. "Composing letters with a simulated listening typewriter". In: *Communications of the ACM* Vol. 26, No. 4 (1983), pp. 295–308. DOI: 10.1145/2163.358100.

[263] Victoria Uren, Yuangui Lei, Vanessa Lopez, Haiming Liu, Enrico Motta, and Marina Giordanino. "The usability of semantic search tools: a review". In: *The Knowledge Engineering Review* Vol. 22, No. 4 (2007), pp. 361–377. DOI: 10.1017/S0269888907001233.

[264] Kemafor Anyanwu, Angela Maduko, and Amit P. Sheth. "SemRank: ranking complex relationship search results on the semantic web". In: *Proceedings of the 14th International Conference on World Wide Web*. ACM, 2005, pp. 117–127. DOI: 10.1145/1060745.1060766.

[265] *Word Economic Forum How much data is generated each day?* Accessed: 2021-07-30. URL: https://www.weforum.org/agenda/2019/04/how-much-data-is-generated-each-day-cf4bddf29f/.

[266] Robert Goecke. "The Evolution of Online Booking Systems". In: *Handbook of e-Tourism*. Springer International Publishing, 2020, pp. 1–25. ISBN: 978-3-030-05324-6. DOI: 10.1007/978-3-030-05324-6_27-1.

[267] Aditya Deshpande, Alisha Shahane, Darshana Gadre, Mrunmayi Deshpande, and Prachi Joshi. "A survey of various chatbot implementation techniques". In: vol. XI. 2017. DOI: 10.1145/792550.792552.

[268] Aysem Diker Vanberg. "From Archie to Google: Search engine providers and emergent challenges in relation to EU competition

law". In: *European Journal of Law and Technology* Vol. 3, No. 1 (2012).

[269] Alan Poulter. "The design of World Wide Web search engines: a critical review". In: *Program* Vol. 31, No. 2 (1997), pp. 131–145.

[270] Arno Breitfuss, Karen Errou, Anelia Kurteva, and Anna Fensel. "Representing emotions with knowledge graphs for movie recommendations". In: *Future Generation Computer Systems* (2021). DOI: 10.1016/j.future.2021.06.001.

[271] Denis Lukovnikov, Asja Fischer, and Jens Lehmann. "Pretrained transformers for simple question answering over knowledge graphs". In: *Proceedings of the International Semantic Web Conference*. Springer. 2019, pp. 470–486. DOI: 10.1007/978-3-030-30793-6_27.

[272] Yoon Kim. "Convolutional Neural Networks for Sentence Classification". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 2014, pp. 1746–1751. DOI: 10.3115/v1/D14-1181.

[273] Binxuan Huang, Han Wang, Tong Wang, Yue Liu, and Yang Liu. "Entity Linking for Short Text Using Structured Knowledge Graph via Multi-Grained Text Matching." In: *Interspeech*. 2020, pp. 4178–4182. DOI: 10.21437/interspeech.2020-1934.

[274] Clinton Gormley and Zachary Tong. *Elasticsearch: the definitive guide: a distributed real-time search and analytics engine*. O'Reilly Media, Inc., 2015. ISBN: 978-1-4493-5854-9.

[275] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. "Bag of tricks for efficient text classification". In: *ArXiv* (2016). DOI: 10.48550/arXiv.1607.01759.

[276] Sherzod Hakimov, Soufian Jebbara, and Philipp Cimiano. "Evaluating architectural choices for deep learning approaches for question answering over knowledge bases". In: *Proceedings of the IEEE 13th International Conference on Semantic Computing (ICSC)*. IEEE. 2019, pp. 110–113. DOI: 10.1109/ICOSC.2019.8665496.

[277] Ram Ganesan Athreya, Axel Cyrille Ngonga Ngomo, and Ricardo Usbeck. "Enhancing Community Interactions with Data-Driven Chatbots–The DBpedia Chatbot". In: *Proceedings of the 2018 Web Conference*. 2018, pp. 143–146. DOI: 10.1145/3184558.3186964.

[278] Paolo Ferragina and Ugo Scaiella. "Tagme: on-the-fly annotation of short text fragments (by wikipedia entities)". In: *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*. 2010, pp. 1625–1628. DOI: 10.1145/1871437.1871689.

[279] Dimitrios Lyras, Kyriakos Sgarbas, and Nikolaos Fakotakis. "Using the levenshtein edit distance for automatic lemmatization: A case study for modern greek and english". In: *Proceedings of the 19th IEEE International Conference on Tools with Artificial Intelligence (ICTAI)*. Vol. 2. IEEE. 2007, pp. 428–435. DOI: 10.1109/ICTAI.2007.41.

[280] Yanlin Feng, Xinyue Chen, Bill Yuchen Lin, Peifeng Wang, Jun Yan, and Xiang Ren. "Scalable Multi-Hop Relational Reasoning for Knowledge-Aware Question Answering". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 2020, pp. 1295–1309. DOI: 10.18653/v1/2020.emnlp-main.99.

[281] Xiao Huang, Jingyuan Zhang, Dingcheng Li, and Ping Li. "Knowledge graph embedding based question answering". In: *Proceedings of the 12th ACM International Conference on Web Search and Data Mining*. 2019, pp. 105–113. DOI: 10.1145/3289600.3290956.

[282] Weiguo Zheng, Hong Cheng, Lei Zou, Jeffrey Xu Yu, and Kangfei Zhao. "Natural language question/answering: Let users talk with the knowledge graph". In: *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. 2017, pp. 217–226. DOI: 10.1145/3132847.3132977.

[283] Jiang Zhong, Chen Wang, Qi Li, and Qing Li. "A New Graph-Partitioning Algorithm for Large-Scale Knowledge Graph". In: *Proceedings of the International Conference on Advanced Data Mining and Applications*. Springer. 2018, pp. 434–444. DOI: 10.1007/978-3-030-05090-0_37.

[284] Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia D'amato, Gerard De Melo, Claudio Gutierrez, Sabrina Kirrane, José Emilio Labra Gayo, Roberto Navigli, Sebastian Neumaier, Axel Cyrille Ngonga Ngomo, Axel Polleres, Sabbir Rashid, Anisa Rula, Lukas Schmelzeisen, Juan Sequeda, Steffen Staab, and Antoine Zimmermann. "Knowledge graphs". In: *ACM Computing Surveys* Vol. 54, No. 4 (2021), pp. 1–37. DOI: 10.1145.

[285] Johannes Hoffart, Fabian Suchanek, Klaus Berberich, and Gerhard Weikum. "YAGO2: A spatially and temporally enhanced knowledge base from Wikipedia". In: *Artificial Intelligence* Vol. 194 (2013), pp. 28–61. DOI: 10.1016/j.artint.2012.06.001.

[286] Eleftherios Dimitrakis, Konstantinos Sgontzos, and Yannis Tzitzikas. "A survey on question answering systems over linked data

and documents". In: *Journal of Intelligent Information Systems* Vol. 55, No. 2 (2020), pp. 233–259. DOI: 10.1007/s10844-019-00584-7.

[287] Mikhail Galkin, Priyansh Trivedi, Gaurav Maheshwari, Ricardo Usbeck, and Jens Lehmann. "Message Passing for Hyper-Relational Knowledge Graphs". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2020, pp. 7346–7359. DOI: 10.18653/v1/2020.emnlp-main.596.

[288] Pooja Devi, Ashlesha Gupta, and Ashutosh Dixit. "Comparative study of hits and pagerank link based ranking algorithms". In: *International Journal of Advanced Research in Computer and Communication Engineering* Vol. 3, No. 2 (2014), pp. 5749–5754. DOI: 10.1109/ICCST.2014.7045007.

[289] Yi Yang and Ming Wei Chang. "S-MART: Novel Tree-based Structured Learning Algorithms Applied to Tweet Entity Linking". In: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language*. Association for Computational Linguistics, 2015, pp. 504–513. DOI: 10.3115/v1/P15-1049.

[290] Yanchao Hao, Yuanzhe Zhang, Kang Liu, Shizhu He, Zhanyi Liu, Hua Wu, and Jun Zhao. "An end-to-end model for question answering over knowledge base with cross-attention combining global knowledge". In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*. 2017, pp. 221–231. DOI: 10.18653/v1/P17-1021.

[291] Kun Xu, Siva Reddy, Yansong Feng, Songfang Huang, and Dongyan Zhao. "Question Answering on Freebase via Relation

Extraction and Textual Evidence". In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 2016, pp. 2326–2336. DOI: 10.18653/v1/P16-1220.

[292] Yu Chen, Lingfei Wu, and Mohammed Zaki. "Bidirectional Attentive Memory Networks for Question Answering over Knowledge Bases". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, 2019, pp. 2913–2923. DOI: 10.18653/v1/N19-1299.

[293] Siva Reddy, Mirella Lapata, and Mark Steedman. "Large-scale semantic parsing without question-answer pairs". In: *Transactions of the Association for Computational Linguistics* Vol. 2 (2014), pp. 377–392. DOI: 10.1162/tacl_a_00190.

[294] Mo Yu, Wenpeng Yin, Kazi Saidul Hasan, Cicero Santos, Bing Xiang, and Bowen Zhou. "Improved Neural Relation Detection for Knowledge Base Question Answering". In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 2017, pp. 571–581. DOI: 10.18653/v1/P17-1053.

[295] Vanessa Lopez, Christina Unger, Philipp Cimiano, and Enrico Motta. "Evaluating question answering over linked data". In: *Journal of Web Semantics* Vol. 21 (2013), pp. 3–13. DOI: 10.1016/j.websem.2013.05.006.

[296] Yuyu Zhang, Hanjun Dai, Zornitsa Kozareva, Alexander J. Smola, and Le Song. "Variational Reasoning for Question Answering with Knowledge Graph". In: *Proceedings of the 32nd AAAI Conference*

on Artificial Intelligence and 13th Innovative Applications of Artificial Intelligence Conference and 8th AAAI Symposium on Educational Advances in Artificial Intelligence. AAAI Press, 2018. ISBN: 978-1-57735-800-8.

[297] Arnaldo Pereira, Alina Trifan, Rui Pedro Lopes, and José Luus Oliveira. "Systematic review of question answering over knowledge bases". In: *Institution of Engineering and Technology (IET) Software* Vol. 16 (2021), pp. 1–13. DOI: 10.1049/sfw2.12028.

[298] Reid Andersen, Fan Chung, and Kevin Lang. "Local partitioning for directed graphs using PageRank". In: *International Workshop Proceedings on Algorithms and Models for the Web-Graph*. Springer. 2007, pp. 166–178. DOI: 10.1007/978-3-540-77004-6_13.

[299] Erwin Filtz. "Knowledge Graphs for Analyzing and Searching Legal Data". Accessed: 2022-12-16. PhD dissertation. Vienna University of Economics and Business, 2021. URL: https://aic.ai.wu.ac.at/~polleres/supervised_theses/Erwin_Filtz_PhD_2021.pdf.

[300] Christina Feilmayr and Wolfram Wöß. "An analysis of ontologies and their success factors for application to business". In: *Data & Knowledge Engineering* Vol. 101 (2016), pp. 1–23. DOI: 10.1016/j.datak.2015.11.003.

[301] Pascal Hitzler, Markus Krotzsch, and Sebastian Rudolph. *Foundations of semantic web technologies*. Chapman and Hall/CRC, 2009.

[302] Ho Hoang Hung, Sourav Bhowmick, Ba Quan Truong, Byron Choi, and Shuigeng Zhou. "QUBLE: towards blending interactive visual subgraph search queries on large networks". In: *The*

VLDB Journal Vol. 23, No. 3 (2014), pp. 401–426. DOI: 10.1007/s00778-013-0322-1.

[303] Matteo Lissandrini, Torben Bach Pedersen, Katja Hose, and Davide Mottin. "Knowledge graph exploration: where are we and where are we going?" In: *ACM SIGWEB Newsletter* (2020), pp. 1–8. DOI: 10.1145/3409481.3409485.

[304] Denis Lukovnikov, Asja Fischer, Jens Lehmann, and Sören Auer. "Neural network-based question answering over knowledge graphs on word and character level". In: *Proceedings of the 26th International Conference on World Wide Web*. 2017, pp. 1211–1220. DOI: 10.1145/3038912.3052675.

[305] Wen Tau Yih, Ming Wei Chang, Xiaodong He, and Jianfeng Gao. "Semantic Parsing via Staged Query Graph Generation: Question Answering with Knowledge Base". In: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*. Association for Computational Linguistics, 2015, pp. 1321–1331. DOI: 10.3115/v1/P15-1128.

[306] Amir Pouran Ben Veyseh. "Cross-lingual question answering using common semantic space". In: *Proceedings of TextGraphs-10: the Workshop on Graph-based Methods for Natural Language Processing*. 2016, pp. 15–19. DOI: 10.18653/v1/W16-1403.

[307] Jafar Alzubi, Anand Nayyar, and Akshi Kumar. "Machine learning from theory to algorithms: an overview". In: *Journal of Physics: Conference Series*. Vol. 1142. 1. 2018, p. 012012. DOI: 10.1088/1742-6596/1142/1/012012.