IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 7 June 2022, accepted 21 June 2022, date of publication 29 June 2022, date of current version 7 July 2022.
Digital Object Identifier 10.1109/ACCESS.2022.3187178

RESEARCH ARTICLE

Question Answering Over Knowledge Graphs: A Case Study in Tourism

SAREH AGHAEI[1], ELIE RAAD[2], AND ANNA FENSEL[1,3,4]
[1]Department of Computer Science, Semantic Technology Institute (STI) Innsbruck, University of Innsbruck, 6020 Innsbruck, Austria
[2]WordLift, 00186 Rome, Italy
[3]Wageningen Data Competence Center, Wageningen University and Research, 6708 PB Wageningen, The Netherlands
[4]Chair Group Consumption and Healthy Lifestyles, Wageningen University and Research, 6706 KN Wageningen, The Netherlands
Corresponding author: Sareh Aghaei (sareh.aghaei@sti2.at)

This work was supported in part by the Project WordLiftNG within the Eureka, Eurostars Program with the Austrian Research Promotion Agency (FFG) under Grant 877857; and in part by the Project KI-NET within the Interreg Osterreich-Bayern 2014–2020 Program under Grant AB 292.

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.

INDEX TERMS Knowledge graphs, question answering, semantic parsing, small and medium enterprises.

I. 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 [1], [2]. A Resource Description Framework (RDF) KG [3], which can also be viewed as a labelled graph, is a collection of RDF triples including three fields: subject, predicate and object [4].

Beyond the generic, huge, and open-world KGs with millions or billions of facts (e.g., DBPedia [5]), KGs are increasingly used in business scenarios [6]. 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 [7], [8]. 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 [9]. In the tourism domain, websites use small and medium scaled KGs to manage tourism information for different regions and improve the traveller experience [10]–[14].

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,

The associate editor coordinating the review of this manuscript and approving it for publication was Pasquale De Meo.

[1]SPARQL Protocol and RDF Query Language.

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 systems [15], [16]. 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 [15].

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 [17]. 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 [18]–[20]. 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 [21] 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 II) 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 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.

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

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 II reviews the related works, Section III formulates the problem, Section IV introduces the proposed approach, Section V presents the experiment and evaluation results, Section VI provides the discussion and Section VII contains the conclusion and future work.

II. 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 [22]–[25]. Although these approaches lead to high precision, recall is low due to low coverage of the variety of questions [26]. 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 [27]. 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 [28], 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 [29]–[32] 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 [31] 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 [19]. A structure, namely the syntactic graph, is introduced in [33] 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 [34] 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 [35] which includes 5K questions paired with their SPARQL queries. A comparative study is presented in [36] using the LC-QuAD 2.0 dataset [37] that consists of 30K question-answer pairs. The objective of this study is to compare different classifiers, including random forest classifier [38] and XGBoost classifier, based on different pre-processing techniques, including POS tags, word embeddings and combination of POS tags and word embeddings. Similar to [34], [36], TeBaQA [39] 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 [40]) 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 [34].

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 [34], [36] 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 RDF triples in a semi-automated solution. Then, in contrast to [39], 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]NLIWOD - Natural Language Interfaces for the Web of Data: https://github.com/semantic-systems/NLIWOD

III. 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.

IV. PROPOSED APPROACH

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

A. OVERVIEW

In our work, the proposed approach consists of two phases, including an offline phase and a semantic parsing phase, as shown in Figure 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 [41] 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.

[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.

TABLE 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

B. 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) [42]–[44].

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 2 depicts the workflow of the offline phase, which is explained in detail as follows.

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 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 [35], [45]–[48] 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 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 1. For example, the RDF templates of the unlabelled tree with $n = 2$ are depicted in Figure 3.

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

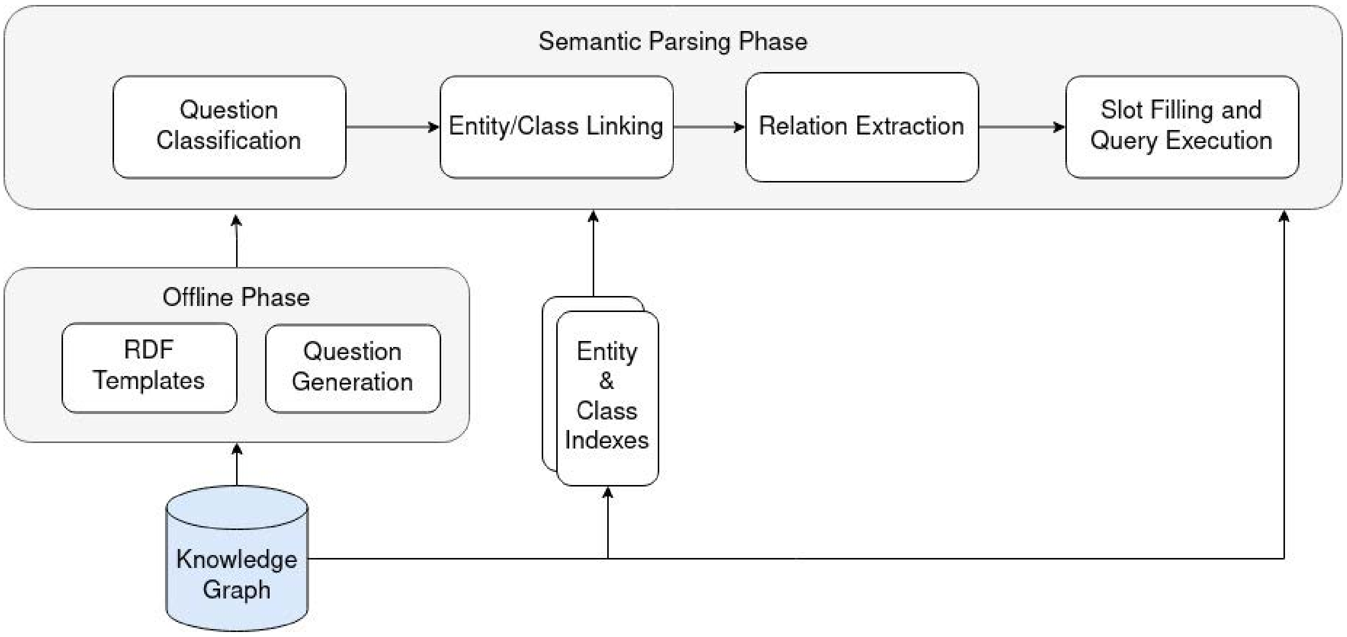


FIGURE 1. **An overview of the proposed approach.**

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 distribution 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").

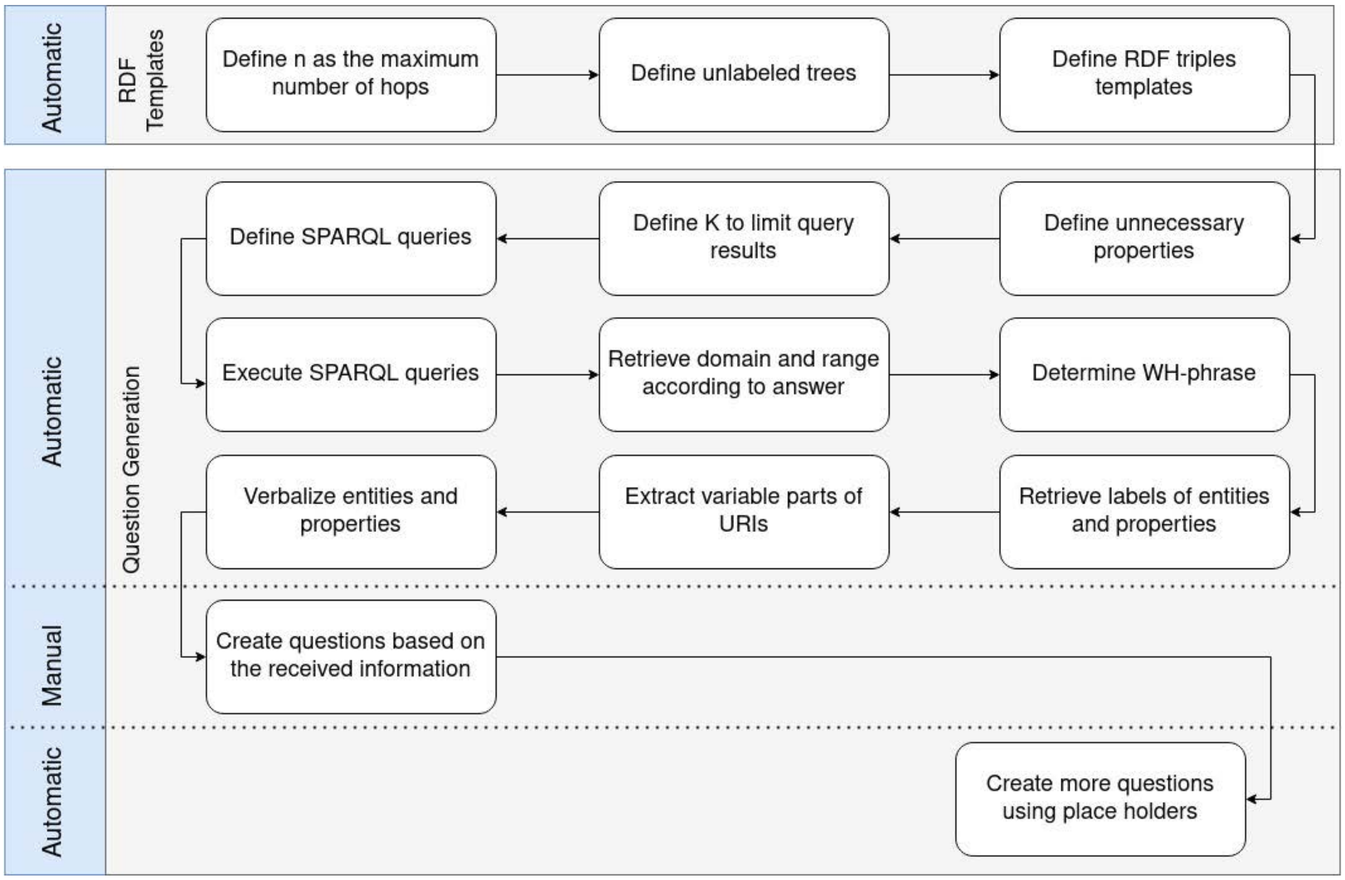
Thus, the information obtained for each row query result includes the verbalized triples, the answer and the WH-word. We employ this information to create a question manually in the next step. Since humans create questions, they are expressed differently due to humans creating questions based on their 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 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>.

C. 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 [28]. Since SPARQL is basically a graph-based query language, an isomorphism can be used to determine the structural equivalence of two SPARQL queries [39]. According to graph isomorphism, two query



Automatic
RDF Templates
Define n as the maximum number of hops
Define unlabeled trees
Define RDF triples templates
Automatic
Question Generation
Define SPARQL queries
Define K to limit query results
Define unnecessary properties
Execute SPARQL queries
Retrieve domain and range according to answer
Determine WH-phrase
Verbalize entities and properties
Extract variable parts of URIs
Retrieve labels of entities and properties
Manual
Create questions based on the received information
Automatic
Create more questions using place holders

FIGURE 2. The workflow of the offline phase.

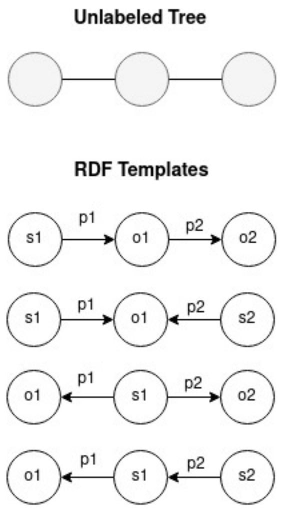


FIGURE 3. Unlabelled tree and RDF templates with 2 hops.

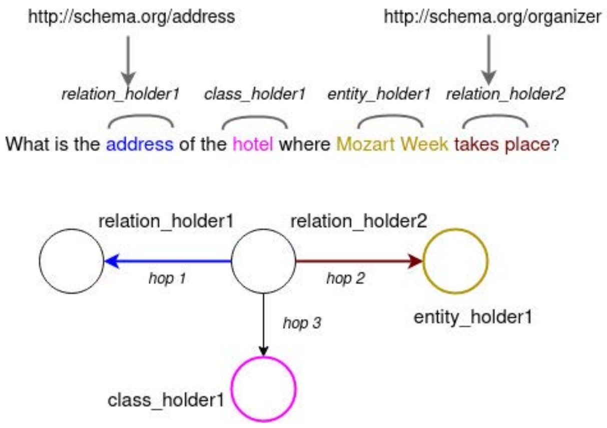


FIGURE 4. An example of collected information of a question.

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 5.

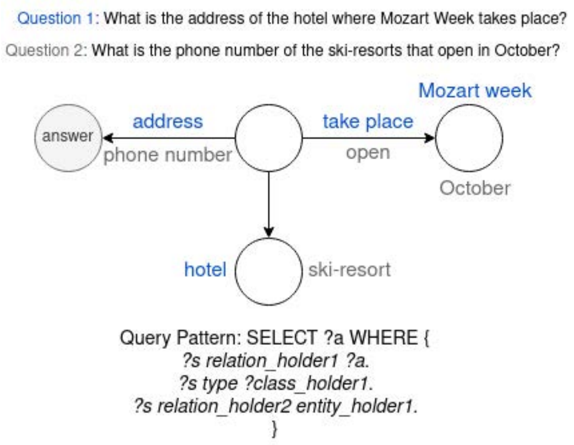

Question 1: What is the address of the hotel where Mozart Week takes place?
Question 2: What is the phone number of the ski-resorts that open in October?
answer
address
phone number
take place
open
Mozart week
October
hotel
ski-resort
Query Pattern: SELECT ?a WHERE {
?s relation_holder1 ?a.
?s type ?class_holder1.
?s relation_holder2 entity_holder1.
}

FIGURE 5. Semantically different questions with similar query pattern.

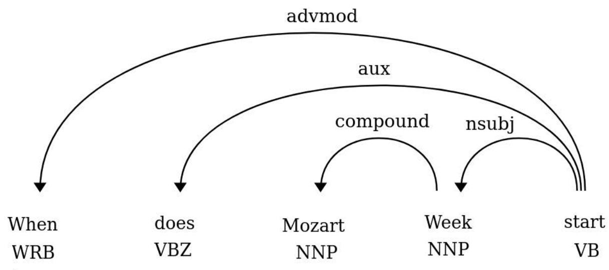


FIGURE 6. An example of dependency parsing.

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 V-B, we show all the basic query patterns used as the question classes in our use-case.

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 models with lower classification accuracy and builds a highly accurate and low false positive model through the constant iteration of the model [41].

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 word>. Considering the question sentence "When does Mozart Week start?", Figure 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>.

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.

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

Figure 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.

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 this end, 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, 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 [49] 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 [50]. 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 [51], 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 [52]. 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.

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 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 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
```

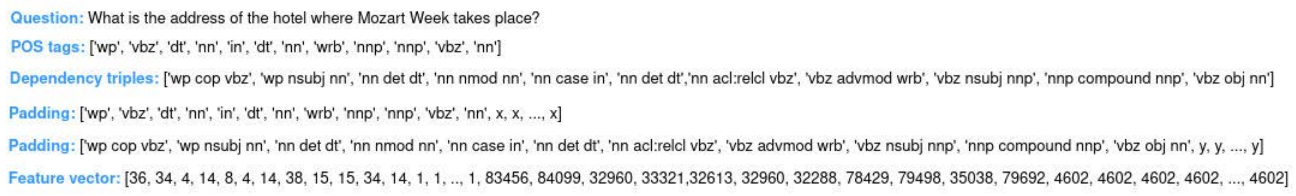
Next, the entities and classes with the highest similarity scores are selected according to the required number of entity holders and class holders in the query pattern.

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 [22]. Since a question sentence can include more than a relation (such as the described questions in Figure 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

[8]Either URI resources or literal resources.
[9]Sentence-Bidirectional Encoder Representations from Transformers.
[10]Bidirectional Encoder Representations from Transformers.



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 7. An example of feature vectors in the question classifier.

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.

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.

V. EXPERIMENTAL STUDY

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

A. 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.

TABLE 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

B. 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 2.

Out of all the possible RDF templates in Table 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, 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 shows the statistics.

To present feature vectors of the questions, the maximum numbers of POS tagging and dependency triples of the

[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. 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

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 5.

C. 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:

$$\text{Accuracy} = \frac{\text{correct predictions}}{\text{total predictions}} \tag{1}$$

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:

$$R = \frac{\text{correctly obtained answers}}{\text{gold standard answers}} \tag{2}$$

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

$$P = \frac{\text{correctly obtained answers}}{\text{returned answers}} \tag{3}$$

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

$$F1 = 2 \times \frac{P \times R}{P + R} \tag{4}$$

D. 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 4 and 5 show the tuned hyper-parameters of XGBoost and MLP classifiers, respectively.

As seen in Table 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 V-B, 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 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 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 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 6, the final accuracy is 0.72, which means 72 percent of questions (127 questions out of a total of 175 questions) can

[14]Rectified Linear Unit.

TABLE 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

TABLE 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 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

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.

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 cannot 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 answers.

TABLE 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 [39]. According to the described features[15] extracted from questions in TeBaQA, we train three

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

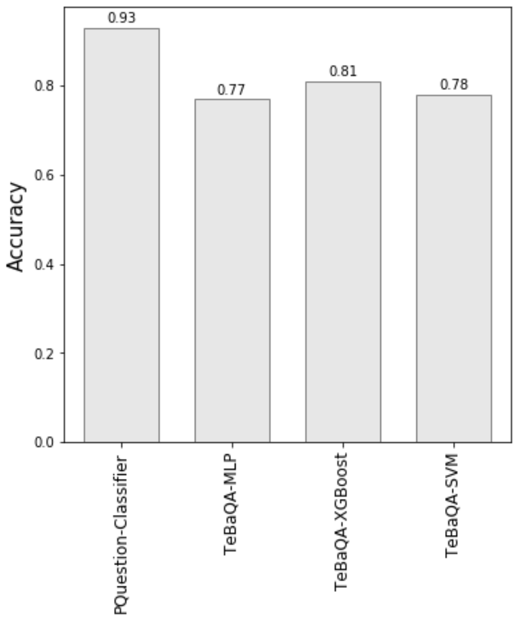


FIGURE 8. Proposed approach versus the state of the art TeBaQA result.

classifiers, including an MLP classifier, XGBoost classifier and support vector machine (SVM) classifier, to identify the query patterns of the questions. Figure 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%.

Note that no single KGQAS will ever be a perfect system for all the steps (question classification, entity and class linking, relation extraction and query execution) on all scales of KGs. For example, our proposed solution for entity linking cannot 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.

VI. 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 V-D, 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 I).
- 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.

VII. CONCLUSION

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 available). 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.

ACKNOWLEDGMENT

The authors would like to thank Tek Raj Chhetri for the insightful discussions. Additionally, they would like to express their gratitude to Albin Ahmeti for providing feedback that aided in the manuscript's improvement. Finally, they thank Jubril Gbolahan Adigun and Mooud Amirkavei for proofreading the manuscript.

REFERENCES

[1] S. Ji, S. Pan, E. Cambria, P. Marttinen, and S. Y. Philip, "A survey on knowledge graphs: Representation, acquisition, and applications," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 2, pp. 494–514, Apr. 2021.

[2] B. Abu-Salih, "Domain-specific knowledge graphs: A survey," *J. Netw. Comput. Appl.*, vol. 185, Jul. 2021, Art. no. 103076.

[3] G. Klyne. (2004). *Resource Description Framework (RDF): Concepts and Abstract Syntax*. [Online]. Available: http://www.w3. org/TR/2004/REC-rdf-concepts-20040210

[4] H. Arnaout and S. Elbassuoni, "Effective searching of RDF knowledge graphs," *J. Web Semantics*, vol. 48, pp. 66–84, Jan. 2018.

[5] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. N. Mendes, S. Hellmann, M. Morsey, P. van Kleef, S. Auer, and C. Bizer, "DBpedia—A large-scale, multilingual knowledge base extracted from Wikipedia," *Semantic Web*, vol. 6, no. 2, pp. 167–195, 2015.

[6] M. Mansfield, V. Tamma, P. Goddard, and F. Coenen, "Capturing expert knowledge for building enterprise SME knowledge graphs," in *Proc. 11th Knowl. Capture Conf.*, Dec. 2021, pp. 129–136.
[7] Z. Akbar, A. Fensel, and D. Fensel, "On generating stories from semantically annotated tourism-related content," in *Proc. OTM Confederated Int. Conf. Meaningful Internet Syst.* Cham, Switzerland: Springer, 2018, pp. 481–497.
[8] A. Fensel, Z. Akbar, E. Kärle, C. Blank, P. Pixner, and A. Gruber, "Knowledge graphs for online marketing and sales of touristic services," *Information*, vol. 11, no. 5, p. 253, May 2020.
[9] E. G. Kalayci, I. G. González, F. Lösch, G. Xiao, A. ul-Mehdi, E. Kharlamov, and D. Calvanese, "Semantic integration of Bosch manufacturing data using virtual knowledge graphs," in *Proc. Int. Semantic Web Conf.* Cham, Switzerland: Springer, 2020, pp. 464–481.
[10] D. Xiao, N. Wang, J. Yu, C. Zhang, and J. Wu, "A practice of tourism knowledge graph construction based on heterogeneous information," in *Proc. China Nat. Conf. Chin. Comput. Linguistics*. Cham, Switzerland: Springer, 2020, pp. 159–173.
[11] C. Lu, P. Laublet, and M. Stankovic, "Travel attractions recommendation with knowledge graphs," in *Proc. Eur. Knowl. Acquisition Workshop*. Cham, Switzerland: Springer, 2016, pp. 416–431.
[12] X. Liang, H. Cao, and W. Zhang, "Knowledge extraction experiment based on tourism knowledge graph Q & A data set," in *Proc. IEEE Int. Conf. Power, Intell. Comput. Syst. (ICPICS)*, Jul. 2020, pp. 828–832.
[13] E. Kärle, U. Şimşek, O. Panasiuk, and D. Fensel, "Building an ecosystem for the tyrolean tourism knowledge graph," in *Proc. Int. Conf. Web Eng.* Cham, Switzerland: Springer, 2018, pp. 260–267.
[14] W. Zhang, T. Gu, W. Sun, Y. Phatpicha, L. Chang, and C. Bin, "Travel attractions recommendation with travel spatial-temporal knowledge graphs," in *Proc. Int. Conf. Pioneering Comput. Scientists, Eng. Educators*. Singapore: Springer, 2018, pp. 213–226.
[15] S. Hu, L. Zou, J. X. Yu, H. Wang, and D. Zhao, "Answering natural language questions by subgraph matching over knowledge graphs," *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 5, pp. 824–837, May 2017.
[16] A. Ait-Mlouk and L. Jiang, "KBot: A knowledge graph based chatbot for natural language understanding over linked data," *IEEE Access*, vol. 8, pp. 149220–149230, 2020.
[17] B. Ojokoh and E. Adebisi, "A review of question answering systems," *J. Web Eng.*, vol. 17, no. 8, pp. 717–758, 2018.
[18] Y. Qiu, Y. Wang, X. Jin, and K. Zhang, "Stepwise reasoning for multi-relation question answering over knowledge graph with weak supervision," in *Proc. 13th Int. Conf. Web Search Data Mining*, Jan. 2020, pp. 474–482.
[19] B. Fu, Y. Qiu, C. Tang, Y. Li, H. Yu, and J. Sun, "A survey on complex question answering over knowledge base: Recent advances and challenges," 2020, *arXiv:2007.13069*.
[20] S. Aghaei and A. Fensel, "Building knowledge subgraphs in question answering over knowledge graphs," in *Proc. 22nd Int. Conf. Web Eng.*, 2022. [Online]. Available: https://icwe2022.webengineering.org/programandsessions/
[21] *Wordlift New Generation*. Accessed: Apr. 10, 2022. [Online]. Available: https://wordlift.io/ng/
[22] C. Unger, L. Bühmann, J. Lehmann, A.-C. N. Ngomo, D. Gerber, and P. Cimiano, "Template-based question answering over RDF data," in *Proc. 21st Int. Conf. World Wide Web*, 2012, pp. 639–648.
[23] W. Zheng, L. Zou, X. Lian, J. X. Yu, S. Song, and D. Zhao, "How to build templates for RDF question/answering: An uncertain graph similarity join approach," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2015, pp. 1809–1824.
[24] W. Zheng, J. X. Yu, L. Zou, and H. Cheng, "Question answering over knowledge graphs: Question understanding via template decomposition," *Proc. VLDB Endowment*, vol. 11, no. 11, pp. 1373–1386, Jul. 2018.
[25] D. Diefenbach, "Question answering over knowledge bases," Ph.D. dissertation, Dept. Sci. Ingénierie Santé, Université de Lyon, Lyon, France, 2018.
[26] W. Cui, Y. Xiao, H. Wang, Y. Song, S.-w. Hwang, and W. Wang, "KBQA: Learning question answering over QA corpora and knowledge bases," 2019, *arXiv:1903.02419*.
[27] S. Ou, C. Orasan, D. Mekhaldi, and L. Hasler, "Automatic question pattern generation for ontology-based question answering," in *Proc. Flairs Conf.*, 2008, pp. 183–188.
[28] H. Bast and E. Haussmann, "More accurate question answering on freebase," in *Proc. 24th ACM Int. Conf. Inf. Knowl. Manage.*, Oct. 2015, pp. 1431–1440.
[29] H. Sun, B. Dhingra, M. Zaheer, K. Mazaitis, R. Salakhutdinov, and W. W. Cohen, "Open domain question answering using early fusion of knowledge bases and text," 2018, *arXiv:1809.00782*.
[30] H. Sun, T. Bedrax-Weiss, and W. W. Cohen, "PullNet: Open domain question answering with iterative retrieval on knowledge bases and text," 2019, *arXiv:1904.09537*.
[31] G. He, Y. Lan, J. Jiang, W. X. Zhao, and J.-R. Wen, "Improving multi-hop knowledge base question answering by learning intermediate supervision signals," in *Proc. 14th ACM Int. Conf. Web Search Data Mining*, 2021, pp. 553–561.
[32] M. Yasunaga, H. Ren, A. Bosselut, P. Liang, and J. Leskovec, "QA-GNN: Reasoning with language models and knowledge graphs for question answering," 2021, *arXiv:2104.06378*.
[33] K. Xu, L. Wu, Z. Wang, M. Yu, L. Chen, and V. Sheinin, "Exploiting rich syntactic information for semantic parsing with graph-to-sequence model," 2018, *arXiv:1808.07624*.
[34] R. G. Athreya, S. K. Bansal, A.-C.-N. Ngomo, and R. Usbeck, "Template-based question answering using recursive neural networks," in *Proc. IEEE 15th Int. Conf. Semantic Comput. (ICSC)*, Jan. 2021, pp. 195–198.
[35] P. Trivedi, G. Maheshwari, M. Dubey, and J. Lehmann, "LC-QuAD: A corpus for complex question answering over knowledge graphs," in *Proc. Int. Semantic Web Conf.* Cham, Switzerland: Springer, 2017, pp. 210–218.
[36] A. K. Dileep, A. Mishra, R. Mehta, S. Uppal, J. Chakraborty, and S. K. Bansal, "Template-based question answering analysis on the LC-QuAD2.0 dataset," in *Proc. IEEE 15th Int. Conf. Semantic Comput. (ICSC)*, Jan. 2021, pp. 443–448.
[37] M. Dubey, D. Banerjee, A. Abdelkawi, and J. Lehmann, "LC-QuAD 2.0: A large dataset for complex question answering over Wikidata and dbpedia," in *Proc. Int. semantic web Conf.* Cham, Switzerland: Springer, 2019, pp. 69–78.
[38] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001.
[39] D. Vollmers, R. Jalota, D. Moussallem, H. Topiwala, A.-C. Ngonga Ngomo, and R. Usbeck, "Knowledge graph question answering using graph-pattern isomorphism," in *Proc. 17th Int. Conf. Semantic Syst. Further Knowl. Graphs*, vol. 53. Amsterdam, The Netherlands: IOS Press, Sep. 2021, p. 103.
[40] P. N. Mendes, M. Jakob, A. García-Silva, and C. Bizer, "Dbpedia spotlight: Shedding light on the web of documents," in *Proc. 7th Int. Conf. Semantic Syst.*, 2011, pp. 1–8.
[41] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2016, pp. 785–794.
[42] H. Gao, L. Wu, P. Hu, and F. Xu, "RDF-to-text generation with graph-augmented structural neural encoders," in *Proc. 29th Int. Joint Conf. Artif. Intell.*, Jul. 2020, pp. 3030–3036.
[43] Y. Chen, L. Wu, and M. J. Zaki, "Reinforcement learning based graph-to-sequence model for natural question generation," 2019, *arXiv:1908.04942*.
[44] K. Xu, L. Wu, Z. Wang, Y. Feng, M. Witbrock, and V. Sheinin, "Graph2Seq: Graph to sequence learning with attention-based neural networks," 2018, *arXiv:1804.00823*.
[45] J. Berant, A. Chou, R. Frostig, and P. Liang, "Semantic parsing on freebase from question-answer pairs," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2013, pp. 1533–1544.
[46] J. Bao, N. Duan, Z. Yan, M. Zhou, and T. Zhao, "Constraint-based question answering with knowledge graph," in *Proc. 26th Int. Conf. Comput. Linguistics, Tech. Papers (COLING)*, 2016, pp. 2503–2514.
[47] W.-T. Yih, M. Richardson, C. Meek, M.-W. Chang, and J. Suh, "The value of semantic parse labeling for knowledge base question answering," in *Proc. 54th Annu. Meeting Assoc. Comput. Linguistics*, vol. 2, 2016, pp. 201–206.
[48] A. Talmor and J. Berant, "The web as a knowledge-base for answering complex questions," 2018, *arXiv:1803.06643*.
[49] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals," *Sov. Phys. Doklady*, vol. 10, no. 8, pp. 707–710, 1966.
[50] L. Yujian and L. Bo, "A normalized Levenshtein distance metric," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 6, pp. 1091–1095, Jun. 2007.
[51] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," 2018, *arXiv:1810.04805*.
[52] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence embeddings using Siamese BERT-networks," 2019, *arXiv:1908.10084*.

SAREH AGHAEI received the M.Sc. degree in computer engineering from the University of Isfahan, Iran, in 2012. She is currently pursuing the Ph.D. degree with the Semantic Technology Institute (STI), Innsbruck, Austria. From 2012 to 2020, she has worked as a Data Analyst at Iran Insurance Company's projects. Since October 2020, she has been a Research Assistant with STI. Her research interests include knowledge graphs and question answering systems.

ELIE RAAD received the Ph.D. degree in computer science from the University of Bourgogne, France, in 2011. After his Ph.D. degree, he held multiple postdoctoral positions at the Memorial University of Newfoundland, Canada; and the University of Bourgogne, where he has worked on designing generative adversarial networks (GANs) and bridging the gap between the semantic web and cognitive science. He is currently working as a Machine Learning Specialist at WordlLift. Prior to this, he has worked as a Data Scientist for different companies, where he was responsible for building knowledge graphs and for generating insights from structured and structured data using artificial intelligence and machine learning algorithms.

ANNA FENSEL was born in Novosibirsk, Russia. She received the university degree in mathematics and computer science from Novosibirsk State University, Russia, in 2003, and the Ph.D. and Habilitation degrees in computer science from the University of Innsbruck, Austria, 2006 and 2018, respectively.

In 2006–2007, she has worked as a Research Fellow at the University of Surrey, U.K., and then as a Senior Researcher at the Telecommunications Research Center Vienna (FTW), Austria, from 2007 to 2013. Afterward, she was an Assistant and then an Associate Professor at the University of Innsbruck. Since 2021, she has been an Associate Professor at Wageningen University and Research, Wageningen, The Netherlands. She has been a co-organizer or a program committee member of more than 100 scientific events, including being in chair roles at top events, such as SEMANTiCS and ESWC, a reviewer for numerous journals, and a Project Proposals Evaluator for funding agencies, such as EU H2020 and FP6, Eureka-Eurostars, and national funding. She is a (co)author of more than 130 refereed publications.