MuHeQA: Zero-shot Question Answering over Multiple and Heterogeneous Knowledge Bases

Carlos Badenes-Olmedo [a,*] and Oscar Corcho [a]

[a] *Ontology Engineering Group, Universidad Politécnica de Madrid, Spain*
E-mails: carlos.badenes@upm.es, oscar.corcho@upm.es
ORCIDs: 0000-0002-2753-9917, 0000-0002-9260-0753

Editors: Bo Fu, California State University Long Beach, USA; Patrick Patrick, Linköping University and University of Gävle, Sweden; Catia Pesquita, Universidade de Lisboa, Portugal
Solicited reviews: Floriano Scioscia, Polytechnic University of Bari, Italy; Takahira Yamaguchi, Keio University, Japan; anonymous reviewer

Abstract. There are two main limitations in most of the existing Knowledge Graph Question Answering (KGQA) algorithms. First, the approaches depend heavily on the structure and cannot be easily adapted to other KGs. Second, the availability and amount of additional domain-specific data in structured or unstructured formats has also proven to be critical in many of these systems. Such dependencies limit the applicability of KGQA systems and make their adoption difficult. A novel algorithm is proposed, MuHeQA, that alleviates both limitations by retrieving the answer from textual content automatically generated from KGs instead of queries over them. This new approach (1) works on one or several KGs simultaneously, (2) does not require training data what makes it is domain-independent, (3) enables the combination of knowledge graphs with unstructured information sources to build the answer, and (4) reduces the dependency on the underlying schema since it does not navigate through structured content but only reads property values. MuHeQA extracts answers from textual summaries created by combining information related to the question from multiple knowledge bases, be them structured or not. Experiments over Wikidata and DBpedia show that our approach achieves comparable performance to other approaches in single-fact questions while being domain and KG independent. Results raise important questions for future work about how the textual content that can be created from knowledge graphs enables answer extraction.

Keywords: question answering, Natural Language Processing, Knowledge Graphs

1. Introduction

Knowledge graphs are now being applied in multiple domains. Knowledge Graph Question Answering (KGQA) has emerged as a way to provide an intuitive mechanism for non-expert users to query

*Corresponding author. E-mail: carlos.badenes@upm.es.

knowledge graphs. KGQA systems do not require specific technical knowledge (e.g., knowledge of SPARQL or Cypher), providing answers in natural language for questions that are also expressed in natural language.

One of the main challenges in the design of KGQA systems is *semantic parsing* [**?**]. In this step, natural language queries (NLQs) are translated into a specific query language (e.g. SPARQL[1] for RDF-based KGs, or Cypher[2] for property graphs). KGQA systems typically use templates with placeholders for relations and entities that must be identified in the original NLQ. Once the template is filled in, the generated query (in SPARQL, Cypher, etc) is evaluated in the system that stores the KG (e.g. Triple store, property graph DB) and the results will provide the answers to the question. Thus, KGQA systems can be reduced to entity and relationship search processes that populate predefined query templates that are commonly identified using supervised machine learning techniques (e.g., supervised classifiers) [1]. This approach is heavily dependent on the data schema to explore the entity relationships and on the availability of training data to create supervised classification models.

Mitigating the dependency of KGQA systems on the underlying graph structure and eliminating the need for training sets is crucial to create cross-cutting solutions more efficiently and with less cost. The first research question addressed in this work is: "*How to extract the answer from a knowledge graph without translating the natural language question into a formal query language?*". Moreover, the dependence on the underlying data schema makes it also difficult for existing KGQA systems to combine knowledge from several graphs to extract a single answer. They typically translate the question into specific queries for each supported KG and obtain multiple answers, rather than combining the knowledge from each KG before drawing a single answer. The combined use of more than one KG to answer a question, even in hybrid solutions that extract answers from both text and KG [2], is another major challenge for KGQA systems. Thus, our second research question is: "*How to create answers that may combine information from multiple knowledge graphs?*". And finally, another relevant challenge in KGQA systems is the limitation in providing explanations for the answers that are obtained. So our third research question is: "*How to provide evidence that supports the response that has been generated by a KGQA system?*".

In this paper, a Multiple and Heterogeneous Question Answering system (MuHeQA) is proposed to address the above research questions. It is a novel KGQA algorithm that works without prior training (i.e. no need to create any learning model from the content of the knowledge graph) and generates answers in natural language without the need to translate the natural language question into a formal query language. Our method combines Extractive Question Answering (EQA) and Natural Language Processing (NLP) techniques with one or more knowledge graphs or other unstructured data sources, to create natural language answers to questions that are formulated in natural language. MuHeQA supports *single fact* questions (i.e. one subject entity, one relation and one object entity), also known as single-hop questions, and can also handle multiple-hop questions (i.e. more than one relationship) by an iterative process after breaking down the question into single-hop questions [3] [4]. Our method provides a brief evidence (i.e. textual content and score) to explain each of the responses that have been obtained. The main contributions, described in this paper, are:

- a **linking method** to find the knowledge graph resources that appear in a natural language question based not only on entities, but also on concepts.
- a **KGQA algorithm** that can extract the answer by combining multiple KGs and other additional unstructured data sources, without prior training.

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

– an **open-source implementation**[3] of the algorithm to facilitate the replication of our experiments and the validation and testing of each of its parts.

This article comprises four additional sections. Section 2 presents the challenges associated with creating QA systems based on knowledge graphs. Our proposal is defined in Section 3, and Section 4 evaluates its performance compared to other state-of-the-art methods on question-answering datasets. The conclusions derived from the results obtained are described in Section 5.

2. Related Work

QA systems commonly divide the process of answering a natural language question into three main tasks [1**?**]: (1) question analysis, (2) information retrieval, and (3) answer extraction. First, they classify the question (e.g. factual, hypothetical, cause-effect), the relevant entities inside the question (e.g., a person or an organization) and the type of expected response (e.g., Boolean, literal, numerical). Second, they process information sources (e.g. databases, documents) based on the previously identified entities. And finally, the answer to the question is provided based on the relevant texts found and the response type.

With the emergence and wide adoption of knowledge graphs, many QA systems have been also adapted to this graph-oriented context, giving rise to KGQA systems. Information is now retrieved from graphs instead of documents or relational databases, and the methods commonly used in QA tasks are adapted to the graph particularities [5]. The *question analysis* task remains similar. However, the entities now do not reside in a text or in a table row in a database, but are resources in the graph with a unique identifier. Entity-linking techniques are then required to discover the graph resources mentioned in a NLQ. The tasks for *information retrieval* and *answer extraction* change radically. The NLQ is usually transformed into a formal query to retrieve the answer from the knowledge graph (e.g. SPARQL, Cypher). This step requires to map natural language expressions as KG relations. In addition, the number of relationships in a question can vary. For example, the single-hop question *'What drug is used to treat schizophrenia?'* has only one relationship (i.e. *drug to treat*), but the multi-hop question *'What are the adverse effects of drugs used to treat schizophrenia?'*) has more than one relationship, namely *'drug to treat'* and *'adverse effects'*, assuming the aforementioned relation instances are part of the vocabulary used by the underlying KG. Note that the number of relation instances in a question also represents, in general, the number of triples that need to be retrieved from the KG to obtain an answer. Recent studies [3] [4] have shown that multi-hop questions can be broken down into single-hop questions and the answer can be extracted through an iterative process of solving simple questions. Thus, the above complex question may be decomposed into *'What are the adverse effects of A?'* and *'is A the drug used to treat schizophrenia?'*, so that the final answer would be obtained by combining techniques based on simple questions. In fact, the challenge of handling single-hop questions has not yet been fully solved, as recent surveys in the area of KGQA have shown [5] [6].

Our work considers questions as single-hop questions and, in case they contain multiple relationships, they should first be decomposed into single-hop questions. There are two general strategies for dealing with single-hop questions based on semantic parsing or information retrieval. On the one hand, semantic parsing techniques depend on predefined examples or rules for transforming input questions into their corresponding logical forms. Falcon 2.0[7] is a heuristic-based approach to the joint detection of entities and relations in Wikidata that creates SPARQL queries based on the entities and relations it identifies. Other methods use rules to support different types of reasoning and estab-

[3]https://github.com/librairy/MuHeQA

lish mappings between questions and the way to extract answers, such as SYGMA [8]. Exist methods that learn patterns semi-automatically [9] or transform the natural language question into a tree or graph representation that is finally converted to a logical form [10]. However, more recent studies such as STaG-QA [11] (Semantic parsing for Transfer and Generalization) tend to work with a generative model that predicts a query skeleton, which includes the query pattern, the different SPARQL operators in it, as well as partial relations based on label semantics. The use of encoder-decoder models to capture the order of words in the query (i.e. sequential models) rather than their structure has also been widely explored recently [12]. They vary according to the decoder they use, either a tree (i.e. seq-to-tree), a sequence (i.e. seq-to-seq), or even combining the tree structure and the sequence in a graph-to-seq representation [1]. The main weakness of these models is that they require a large amount of training data to create supervised models that classify query patterns from the queries. On the other hand, information retrieval-based methods focus on creating and extending subgraphs using the entities identified in the question and the related resources found in the knowledge graph. The nodes in the subgraph that map to entities that do not appear in the question are considered answers, and both questions and candidate answers are often represented in vector spaces where they can be related. Methods vary according to the features used for the representations (e.g. paths between question entities and candidate answers) [6]. Recently, Retrieval Augmented Models (RAGs) have drawn considerable attention from researchers due to its great performance on open domain questions and the simplification of the parameters required for its use, compared to other generative models such as GPT-3 or MegatroneLM. Although these models are based on Wikipedia, the RAG-end2end extension [13] has adapted RAG models to other domains, such as COVID-19, News, and Conversations.

In summary, KGQA systems are mainly focused on improving the accuracy of the query created from the natural language question by collecting related information (i.e. training data, history data). The answer is then extracted from the result of executing the query over the KG or the subgraph generated from it. While this approach has proven to be valid as shown in multiple benchmarks and experiments [1, 6**?**], existing methods show a strong dependence on the underlying data structure, which makes it difficult to reuse QA systems tailored for specific KGs on a different KG. The quality of the results is still highly influenced by how resources are related in the KG, and how accurate the query template classification techniques are. Our approach reduces the dependency on the KG structure by suppressing the translation of the question into a formal query, and enables the use of more than one KG and additional unstructured data sources simultaneously to generate the answer, since it does not require formal language queries to obtain the answer. This novel technique facilitates its reuse in large and general-purpose KGs, such as Wikidata or DBpedia, as well as for small and domain-specific ones.

3. Approach

The MuHeQA method creates natural language answers from natural language questions by using one or more KGs, as well as unstructured data sources (i.e. textual sources in the domain). The response is extracted from a textual summary that is automatically created by combining data retrieved from such multiple sources. This section details each of the steps and illustrates the workflow with a guided example.

MuHeQA is based on three steps, as shown in Figure 1 with a sample question for better presentation. The boxes marked with a dashed line are steps that represent the processing stages: *Summarization*, *Evidence Extraction* and *Answer Composition*. Their main responsibilities are (1) create a context from the question, or sub-questions in the case of decomposing a multi-hop question, (2)

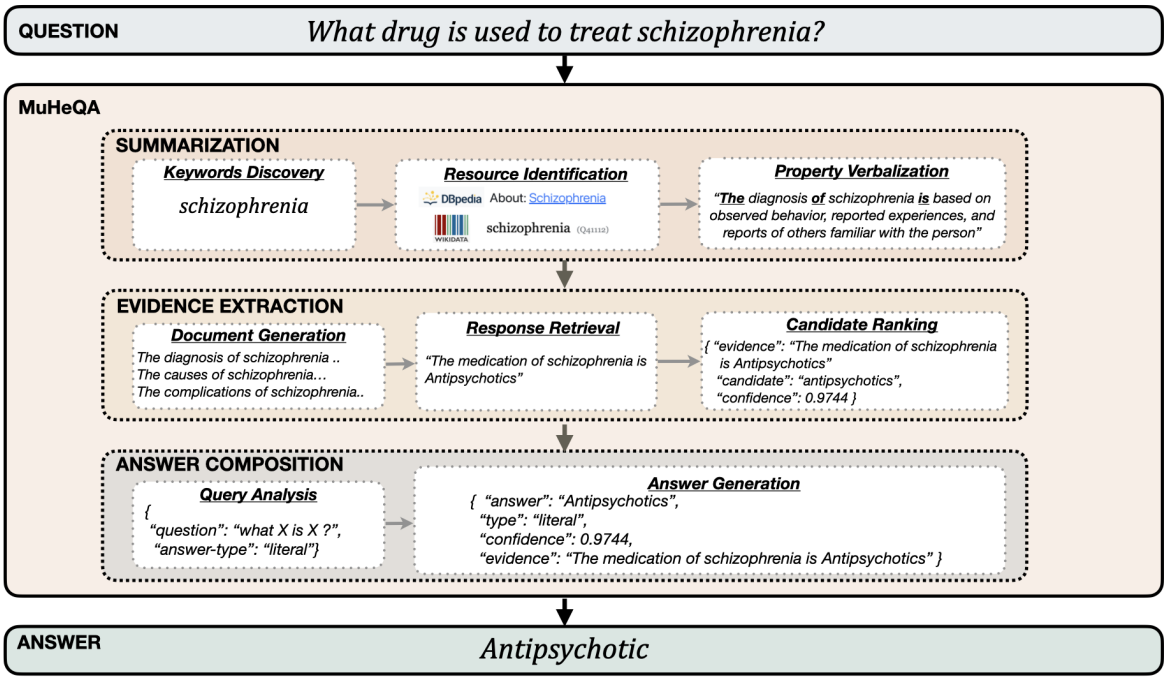

Fig. 1. Tasks involved in the MuHeQA algorithm

extract and sort the candidate responses and (3) compose the final answer. The steps are described in depth below.

3.1. Summarization

Our algorithm can use data retrieved from one or more Knowledge Graphs, $KG : \{kg_1, kg_2, .., kg_m\}$. For a given question q, it first identifies the keywords w_i using common NLP techniques like lemmatization, Part-of-Speech (PoS) tagging and dependency parsing, $W_q : \{w_1, w_2, .., w_n\}$. Keywords not only correspond to proper names, but can also contain nouns, adjectives, or cardinals mentioned in the question (more details in Section 3.1.1). Then, the knowledge graph resources r_j whose label or description contains any of the keywords w_i are considered candidates to generate the text summaries, $R_{ij} : \{r_{11}, r_{12}, r_{21}.., r_{nm}\}$ (more details in Section 3.1.2). A textual summary is created for each resource r_{ij} verbalising the properties in the KG that are related to the question. The summary, expressed as a set of sentences $S_{r_{ij}} : \{s_1, s_2, .., s_t\}$, is composed by joining the sentences that arise when transforming the relevant resource properties, i.e. its value and label, into natural language text. Once all KGs have been explored, the summaries are combined into a single document, $D : \{S_{11}, S_{12}, .., S_{nm}\}$ where n is the number of resources and m is the number of KGs. This textual content can be extended with additional texts retrieved from unstructured data sources using information retrieval techniques (e.g. keyword-based) and is finally used by the next step to extract and sort the evidences related to the answers for the question. Each of these stages is detailed below. This approach based on textual descriptions of the key elements identified in a question from the entity properties in a KG is related to the first research question mentioned above since the natural language question does not need to be translated into a formal query language.

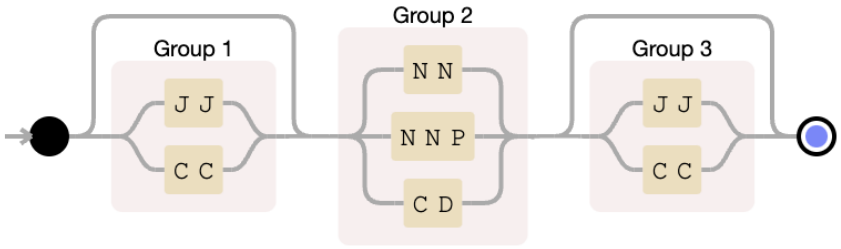

Fig. 2. PoS categories defining a keyword in a question

3.1.1. Keyword Discovery

Our goal in this step is to identify not only named entities but also concepts which will later allow extracting related information to the question from each of the KGs. Unlike other approaches that need to correctly fill the query template with the data extracted from the question, our goal is based on a textual summary that should be as rich as possible to obtain the answer with guarantees. In particular, the focus is on broadening the entity definition where possible (e.g. *'ABC1 protein'* instead of just *'ABC1'* from the question *'what does ABC1 protein transport?'*), and identifying the key concept when no entity is found (e.g. *'overnight delivery'* from the question *'in which country was overnight delivery filmed in?'*). Our method is based on the PoS tags to discover keywords in a query. It uses the 'flair/pos-english' part-of-speech tagging model for English that ships with Flair[14] at Hugging Face[4], and organizes the PoS categories in levels of relevance to guide the keyword composition. At the first level are the categories that most commonly define an entity, such as Noun (NN), Proper Noun (NNP) and Cardinal Number (CD). At the second level are the categories that may characterise the entity, such as Adjective (JJ) and Coordinating Conjunction (CC). And the last level establishes the categories that invalidate the keyword to represent a single entity, such as preposition or subordinating conjunction (IN), Wh-determiner (WDT), verb (VBx) or particle (RP). PoS labels are assigned sequentially as a state machine (see Fig 2) and keywords emerge when terms from the first level are together or next to some from the second level, but not from the third level. In the previous question, *'in which country was overnight delivery filmed in?'*, the keywords *'country'* and *'overnight delivery'* are identified from the labelling: *in (IN) which (WDT) country (NN) was (VBD) overnight (JJ) delivery (NN) filmed (VBN) in (RP) ?*.

3.1.2. Resource Identification

Once the keywords are identified, they have to be linked to available resources of the Knowledge Graphs to obtain related information that may help answer the question. As before, existing approaches are mainly focused on accurately identifying the resources to fit the query template. In our case, resource candidates should be prioritized not only for their textual similarity to the keyword, but also for their relationship to the question. The creation of vector spaces where each resource is represented by its labels [6] is discarded since one of our assumptions is to avoid the creation of supervised models that perform specific classification tasks over the KG (i.e. prior training). Our proposal does not require training datasets since it performs textual searches based on the terms identified in the query using an inverse index of the labels associated with the resources. At this point it should be noted that the more descriptive the property labels in the KG are, the better the text-based search will perform. A rank strategy for the resource candidates based on three types of

[4]https://huggingface.co/flair/pos-english

textual similarity is defined: (1) *at the name level*; (2) *at the property level*; and (3) *at the description level*. Equation 1 states the relevance of a KG resource (res) based on the semantic similarity ($sim(x, y)$) between the given question (q) and the resource name (res_{name}), the resource description (res_{desc}) and the resource property labels (res_{prop_i}). The list of resource candidates is ordered from highest to lowest relevance. Different strategies to select the most relevant ones can be considered.

$$Relevance_{res} = \frac{1}{n}\left(sim(res_{name}, q) + sim(res_{desc}, q) + \sum_{i=1}^{n-2} sim(res_{prop_i}, q) \right) \quad (1)$$

As already pointed out, unlike most existing approaches, the NLQ is not translated into a formal query (e.g. SPARQL) to retrieve the answer. Instead, the available properties in the KGs are queried for the most relevant resources mentioned in the query using the formal query language accepted by the underlying source (SPARQL, Cypher, ..). This step is the only one anchored to the knowledge graph, and we minimize its dependency with the data schema since we only need to explore the values and labels of its properties, instead of traversing the relationships. KGs are organized by resource properties that can be RDF triples, relational table columns or facets. For RDF-based knowledge graphs, i.e. DBPedia or Wikidata, a unique SPARQL query (Figure 3) is used to retrieve all the related information. These queries along with the rest of source code of the algorithm are publicy available for reuse.

The semantic similarity between the resource (i.e. name, property labels and description) and the query is based on the cosine similarity of their vector representations created with a sequence-to-sequence language model. Specifically, it is a sentence-transformers model[5] created from the distilled version of the RoBERTa-base model and fine-tuned on a 1B sentence pairs dataset that maps sentences to a 768 dimensional dense vector space. This metric was considered because in the next step the evidence will be extracted using techniques based on the same representation model. An example about how it works on Wikidata is shown below. The most relevant resource candidates at the *name level* related to the keyword '*Carlos Gomez*' identified in the question '*What position does Carlos Gomez play?*' are: the researchers 'Carlos Gomez'(Q89898891), 'Carlos Gomez' (Q51944192), 'Carlos A Gomez' (Q91676432) and 'Carlos M Gomez' (Q40124092); the american actor 'Carlos Gomez' (Q949506); the mexican footballer 'Carlos Gomez' (Q203210); the Dominican Republic baseball player 'Carlos Gomez' (Q2747238); and the Chilean association of football players born in 1992 'Carlos Gomez' (Q5750557). At the *property level*, i.e. comparing the cosine similarity of the resource property labels with the question, the list of most relevant candidates is reduced to the footballer 'Carlos Gomez' (Q203210) and the baseball player 'Carlos Gomez' (Q2747238). Finally, at the *description level*, i.e. the cosine similarity of the resource description with the question is added, the list is reduced to the footballer 'Carlos Gomez' (Q203210). This selective strategy based only on the best relevant scorers can be varied by manually setting the relevant top n (See section 4 for more details).

In case of unstructured data sources, the resource identification step corresponds to the search for sentences or paragraphs where the keyword identified in the previous step is mentioned. More elaborate strategies could be considered, for example by selecting texts that contain terms related to the keyword, either syntactically or semantically.

3.1.3. Property Verbalization

At this point the property-based comparisons from the previous step are reused to create a textual summary using only the most relevant properties (i.e. labels closest to the question). For each property, its value is obtained from the KG and expressed in a sentence. Although there are methods

[5] https://huggingface.co/sentence-transformers/all-distilroberta-v1

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX wd: <http://www.wikidata.org/entity/>
SELECT distinct ?prop ?propLabel
WHERE
{
  { wd:ENTITY ?a ?b }
        union
        { ?s ?a wd:ENTITY } .

  SERVICE wikibase:label { bd:serviceParam wikibase:language "en". }
  ?prop wikibase:directClaim ?a .
}
LIMIT 250
```

(a) Wikidata

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dbr: <http://dbpedia.org/resource/>
select distinct ?property ?label {
    { <http://dbpedia.org/resource/ENTITY> ?property ?o }
    union
    { ?s ?property <http://dbpedia.org/resource/ENTITY> }

    optional {
      ?property rdfs:label ?label .
      filter langMatches(lang(?label), 'en')
    }
    filter(regex(?property, "property", "i" ))
}
LIMIT 250
```

(b) DBpedia

Fig. 3. SPARQL queries used to extract the properties of a KG resource

that are able to verbalise a relation between a value and an entity through a property, e.g. TekGen [15], the triple-to-text [16] method or UniK-QA [17], the source code is either not available or not currently operational for our approach so a basic solution has been developed for verbalising triples based on applying the following rule: 'The *<property>* of *<entity>* is *<value>*' (e.g. *The duration of schizophrenia is chronic*). Natural language phrases are then created from each property-value available in the knowledge graph that is related to the entity. Labels described in natural language associated with each resource are used: *wikibase:label* in Wikidata and *rdfs:label* in DBpedia. All these phrases make up the summary.

For example, an excerpt from the textual summary created from Wikidata for the question '*What position does Carlos Gomez play?*' is as follows: "*The position played on team / speciality of Carlos Gómez is defender. The given name of Carlos Gómez is Carlos. The member of sports team of Carlos Gómez is Atlético Potosino. The member of sports team of Carlos Gómez is Club León. The member of sports team of Carlos Gómez is Puebla F.C. The family name of Carlos Gómez is Gómez. The occupation of Carlos Gómez is association football player*". The order of appearance of the properties in the summary shows their closeness to the question. The former are more closely related than the latter. (i.e. 'position played on team / speciality', 'given name', etc).

3.2. Evidence Extraction

The objective of this step is to obtain evidences E to answer the question q from the textual summaries previously created in document D. An evidence is an expression (i.e. response candidate) accompanied by one or more phrases that support it, and a score that measures its confidence. Extractive question-answering (EQA) techniques [18] were used to create evidences. EQA is a natural language processing task based on language models that retrieves a short snippet from a context in order to answer a natural language question. A score is attached to the evidence as a value derived from the relevance obtained in the previous step (see equation 1) and the confidence when extracting the response. This approach based on extractive question-answering techniques addresses the second research question proposed in this work, since it supports one or multiple knowledge graphs to create the textual summary from which evidences are extracted.

3.2.1. Document Generation

The verbalisations of the properties in a single text document were combined. If additional unstructured data sources (e.g. texts) are available, this step involves the addition of the sentences or paragraphs where the keywords are mentioned in those data sources. It should be noted that language models limit the amount of text that they can process in each operation, so the document is splitted into smaller parts (no more than 512 words) in order to process them considering sentences as the minimum unit that cannot be broken.

3.2.2. Response Retrieval

A text fragment (i.e. an evidence) from each textual part is extracted using EQA techniques, with the corresponding confidence. The EQA task consists in completing a text sequence (i.e. context) using a language model. The sequence is created by joining the textual content and the question. The model then infers the most likely text that would continue the sequence. This new text is the evidence to discover the answer to the question. Recent methods have even supported multilingual inferences [19], but the use of English language models based on bidirectional encoder representations from transformers (BERT) [20] was preferred to better understand its behavior with the text that is automatically generated from knowledge bases. Specifically, a general-domain language model[6] trained on the SQUAD2.0 dataset [21] is combined with a more specific clinical-domain language model[7] trained on a Covid-19 question answering dataset [22]. The extractions from the models was chained and the evidence was ordered according to its confidence score.

3.2.3. Candidate Ranking

From all the evidences found in the previous step, the one that offers the highest confidence was chosen, although in other tasks a ranked list of potential answers is may also offered. For example, from the textual summary previously created for the question '*What position does Carlos Gomez play?*', the most relevant evidence is 'defender' supported by the sentence: '*The position played on team / speciality of Carlos Gómez is defender*' , with the highest confidence score, equal to 0.99.

3.3. Answer Composition

The last step uses the evidence to create the answer from the analysis of the question. In some cases the evidence does not exactly answer the question. For example, the answer to the question '*How many pharmaceutical products contain the active ingredient Paracetamol?*' is not '*Ofirmev, Tylenol, Midol Long Lasting Relief, Mapap and Tachipirina*', but '*5*'. This step is composed of two sub-steps: query analysis and answer generation.

3.3.1. Query Analysis

The type of the expected answer based on the user question is required. Most of the existing solutions consider domain-specific types, because they need to filter the resources they return in the response. However our approach works also with natural language in the response, so it is sufficient to distinguish the high-level answer categories: literal, numerical and boolean. *Boolean questions* (also referred to as *Confirmation* questions) only admit 'yes' or 'no' as answer (e.g. "Is chloroquine authorized by the European Medicines Agency?"). *Numerical questions* have a number as an answer (e.g. "How many types of intracavitary chemotherapy exist?"). And *literal questions* have a textual value as answer, which can be a string or date (e.g. "What side effects does abacavir commonly report?"). Answer type prediction was approached as a classification problem where each question is a sequence of words, so an existing fine-tuned BERT model trained with general-domain questions for category prediction [23] was used.

3.3.2. Answer Generation

In the case that the type of the response is numeric or boolean, a post-processing of the response is required. For quantities a special character (usually comma) is assumed as separator, and the number of elements in the candidate response is counted. Note that given our approach a listing of the responses is may also provided. In the boolean case it is considered true when the confidence is

[6] https://huggingface.co/deepset/roberta-base-squad2
[7] https://huggingface.co/shaina/covid_qa_mpnet

above a threshold. In other case the answer will be created directly by joining the information about evidence, confidence and type of the answer.

This approach to provide not only the answer but also the text from which it is taken addresses our third research question ("*How to provide evidence that supports the response that has been generated by a KGQA system?*"), since it creates an evidence for the response in the form of a sentence that contextualise it together with a numerical value expressing confidence.

4. Experiment Setup and Discussion of Results

This section describes the experiments, including the datasets and baselines, used to evaluate the main contributions of MuHeQA, and reports the results of a comparison between the proposed approach and the baseline systems. The source code of the algorithm, the experiments and the datasets used are publicly available [8]. In addition, MuHeQA is being used in the Drugs4Covid[9] platform [24] to provide a question-answering interface that unifies the knowledge retrieved from both scientific publications collected in the CORD-19 corpus and the Wikidata and DBpedia knowledge graphs. The web interface is available at https://drugs4covid.oeg.fi.upm.es/services/bio-qa.

4.1. Benchmark

One of the differentiating characteristics of MuHeQA is that it allows combining multiple KGs and additional unstructured data sources, providing answers for simple (a.k.a. single-hop) questions in natural language. Existing datasets that have been used for evaluating KGQA systems usually target one knowledge graph, which means that the structure of the answers is dictated by the conceptual organization of the particular knowledge graph. Moreover, natural language answers are required instead of responses expressed as SPARQL queries (e.g. WebQuestions [25]). As a restriction, our approach only works with single-hop questions instead of multiple-hop questions (e.g. LC-QuAD 2.0 [26], VQuAnDA [27]).

The SimpleQuestions dataset [28] has emerged as the de facto benchmark for evaluating simple questions over knowledge graphs. It focuses on questions that can be answered via the lookup of a single fact (i.e. triple). The dataset gained great popularity with researchers due to its much larger size (more than 100K questions) but unfortunately, Google shut down Freebase in 2015. A final snapshot of the knowledge graph is still available online for download, but the associated APIs are no longer available. The benchmark was then mapped to Wikidata [29] in the SimpleQuestionsWikidata[10] dataset, and to DBpedia[30] in the SimpleDBpediaQA[11] dataset. They mapped the Freebase triples associated with the questions to Wikidata and Dbpedia triples, but not every translated triple is required to exist in the current edition of Wikidata or DBpedia. In addition, the answers are defined as resources, not natural language expressions. The reason is that this benchmark was created to evaluate systems based on SPARQL queries, not systems that generate natural language responses.

We used both the SimpleDBpediaQA and SimpleQuestionsWikidata datasets in our evaluations, and created the natural language answers from the SPARQL queries that they propose. As our approach does not require a training set, we have focused only on the test sets of both benchmarks. A total of 3,667 questions define our evaluation set. This evaluation set is available as part of our additional material[12].

[8] https://github.com/librairy/MuHeQA
[9] https://drugs4covid.oeg.fi.upm.es/
[10] https://github.com/askplatypus/wikidata-simplequestions
[11] https://github.com/castorini/SimpleDBpediaQA
[12] https://github.com/librairy/MuHeQA

4.2. Evaluations

As described above, our method provides one or more natural language answers to natural language questions. The answers contain the natural language text, the evidence (in the form of a sentence or sentences from which the answer was obtained), and a numerical value representing the confidence. The algorithm internally sorts the answers based on that confidence value, from highest to lowest, and finally selects one or more answers as valid. To better understand the performance of our algorithm, we have evaluated five different configurations that vary in the way of selecting the answers (see tables 2 and 3): '*best*' considers as valid only the answers with the highest score (i.e. one or several if they match in score); '*all*' considers all the answers as valid; '*top1*' considers as valid only the first answer (i.e. even if it matches in score with the second one), '*top2*' admits the first and second one, and '*top3*' selects the three most relevant answers.

In order to evaluate the quality of the answers we have applied the most commonly used metric in KGQA based on precision, recall and f-Measure (F1). In particular, we distinguish between **macro-average**, that computes the arithmetic mean of all scores, and **micro-average**, that computes a global average score by counting the sums of the True Positives (TP), False Negatives (FN), and False Positives (FP).

4.3. Results

The performance of the proposed algorithm has been evaluated for solving the main three tasks in KGQA upon receiving a natural language question: (1) identification of keywords, (2) discovery of related resources in the KG and, finally, (3) generation of valid answers, i.e. the behaviour as a whole. Additionally, since the algorithm also supports knowledge based on documents, it has been evaluated to provide answers from a set of text passages.

4.3.1. Keywords in a Question

As described in Section 3.1.1, our method identifies the entities mentioned in a question along with the relevant terms discovered using PoS annotations. The method used is a standard part-of-speech tagging model for English that ships with Flair[14]. The performance of our method (i.e. $MuHeQA_{keyword}$) has been compared with state-of-the-art approaches based on the language models FLERT [31], BERT [20] and RoBERTA [32] for named entity recognition. Specifically, the implementations of the models that have been considered are those fine-tuned from the English version of the standard CoNLL-2003 Named Entity Recognition dataset, and that are available on the HuggingFace platform: 'dslim/bert-base-NER' based on BERT, 'Jean-Baptiste/roberta-large-ner-english' based on RoBERTA and 'flair/ner-english-large' based on FLERT. The evaluation consists of comparing the entities previously identified in questions retrieved from QA benchmarks with those identified by the method to be evaluated. Since in real QA environments the questions are not always well written (e.g. they may omit question marks, or do not use capital letters to name entities), the WikidataQA dataset [33] that has grammatically valid texts (e.g. *Who is the author of* ***One Piece****?*) has not only been used, but also the SimpleQuestions dataset that is more flexible and can omit punctuation marks, capital letters, etc. (e.g. *what position does* ***pee wee reese*** *play in baseball*).

Table 1 shows the results of the analysis. The highest precision is obviously achieved by the language models that have been fine-tuned to solve NER tasks, but at the cost of drastically reducing the coverage of entities. In KGQA systems this is a risk, since if no entity or key concept is identi-

[13] https://huggingface.co/flair/ner-english-large
[14] https://huggingface.co/dslim/bert-base-NER
[15] https://huggingface.co/Jean-Baptiste/roberta-large-ner-english

Table 1

Performance identifing keywords in a question

System	SimpleQuestions [28]						WikidataQA [33]					
	micro			macro			micro			macro		
	P	*R*	*F1*	*P*	*R*	*F1*	*P*	*R*	*F1*	*P*	*R*	*F1*
MuHeQA$_{keywords}$	64.76	74.91	**69.47**	70.71	74.91	**72.09**	70.89	63.75	**67.13**	72.50	65.91	**66.73**
FLERT$_{NER}$ [13]	77.92	53.98	63.78	52.33	53.98	52.87	82.58	37.58	51.85	47.00	37.91	40.79
BERT$_{NER}$ [14]	13.96	2.66	4.46	2.59	2.66	2.61	75.00	34.22	47.00	43.00	34.41	37.29
RoBERTA$_{NER}$ [15]	71.80	62.20	66.65	60.03	62.19	60.74	85.50	39.59	54.12	49.50	39.91	42.95

fied in a question, it would remain unanswered. In addition, language models are also very sensitive to the grammatical characteristics of texts, as evidenced by their lower performance in the *SimpleQuestions* dataset where the texts do not use capital letters in the entities. Our approach based on the combination of entities with the key concepts inferred by PoS categories shows a better performance in both scenarios, where the recall is improved without severely penalizing the precision, thus improving the performance of the algorithm in absolute terms (i.e. f1-score).

4.3.2. Linking to KG Resources

As described in Section 3.1.2, our method discovers relevant resources in a Knowledge Graph from the keywords identified in a question. It performs a textual search based on the terms in the keyword. The resources found in the KG are considered as candidates and, depending on the ranking criteria, will be more or less relevant to the question. In order to better measure the performance several configurations were proposed. $MuHeQA_{all}$ selects all candidates since it considers them to be equally relevant. $MuHeQA_{best}$ selects only the best candidate or candidates (i.e. highest relevance score) after ranking them on the basis of equation 1. $MuHeQA_{topN}$ ranks as before, but only selects the N best candidates.

The behavior of our linking method has also been evaluated on DBpedia and Wikidata. The textual searches were performed using the *wbsearchentities* action[16] in Wikidata, and the Lookup service[17] in DBpedia. They are services that use an inverse index to find a resource from the text of a label (e.g. name or description). In any other KG, an inverse index could be easily created from a resource label (e.g. name or description) to provide a similar service. This avoids any need for a training dataset, since any knowledge graph, whether large or small, general or domain-specific, can offer a search service based on labels. The only requirement is that there is at least one label for each resource and it is recommended that it is sufficiently verbose to find it.

Table 2

Performance when discovering Wikidata resources

System	micro			macro		
	precision	*recall*	*f1*	*precision*	*recall*	*f1*
MuHeQA$_{all}$	8.24	86.07	15.05	12.28	82.83	18.89
MuHeQA$_{best}$	43.68	53.89	48.25	49.64	52.06	50.33
MuHeQA$_{top1}$	**57.53**	**56.00**	**56.75**	**57.50**	**57.01**	**57.17**
MuHeQA$_{top2}$	54.21	59.21	56.60	55.82	58.58	56.87
MuHeQA$_{top3}$	46.73	61.18	52.99	51.81	60.56	54.98
spaCy$_{EL}$	45.89	63.33	53.22	52.14	63.69	54.65

[16] https://www.wikidata.org/w/api.php?action=help&modules=wbsearchentities

[17] https://lookup.dbpedia.org/api/search

The evaluation consists of comparing the resources found by our method based on a Natural Language question and the keywords previously identified. The SimpleQuestions dataset, previously processed to work in Wikidata and DBpedia, is used in this evaluation as it contains the keyword (i.e. entity name) and the KG resource for each question. The performance of our methods was compared with other existing solutions for linking resources in Wikidata (see Table 2) and DBpedia (see Table 3). Specifically, the *spaCy Entity Linker*[18] module was used to analyze performance in Wikidata, and DBpedia Spotlight[19] to make the analogous in DBpedia. Again, performance was measured in terms of (macro and micro) precision, recall and f-measure.

Table 3
Performance when discovering DBpedia resources

System	**micro**			**macro**		
	precision	*recall*	*f1*	*precision*	*recall*	*f1*
MuHeQA$_{all}$	12.13	95.03	21.52	21.07	95.03	29.62
MuHeQA$_{best}$	94.44	91.29	92.84	91.20	91.29	91.23
MuHeQA$_{top1}$	**95.22**	**91.35**	**93.24**	**91.35**	**91.35**	**91.35**
MuHeQA$_{top2}$	87.76	90.86	89.28	88.89	90.86	89.55
MuHeQA$_{top3}$	82.37	90.99	86.47	86.43	90.99	87.93
DBpedia Spotlight [34]	54.85	63.56	58.89	54.87	63.55	57.67

Regardless of the Knowledge Graph, it seems that the most promising approach is the one that considers only the first candidate (i.e. $MuHeQA_{Top1}$). It makes sense since the questions used in the evaluation are single fact questions and therefore only have one entity and one resource associated to them. The poor performance of the ranking-based approach on Wikidata (i.e. $MuHeQA_{best}$) is striking. It is not due to inability to discriminate the most relevant resources, because the recall is also low. We suspect that the reason is that the property labels in Wikidata are not as descriptive as in DBpedia, since we use them to measure their closeness to the question and help to select a candidate resource. And this is especially critical when the keyword, i.e. the label in the KG, is not specific enough. For example, given the keyword '*Myocardial infarction*' identified in the question '*Which Swiss conductor's cause of death is myocardial infarction?*', our $MuHeQA_{best}$ method linked it successfully to the *Q12152* resource in Wikidata (i.e. myocardial infarction as an interruption of blood supply to a part of the heart) and to the '*Myocardial_infarction*' and '*Cardiovascular_disease*' resources in DBpedia. On the other hand, given the keyword '*the medic*' identified in the question '*what language is spoken in the medic*', the method wrongly considers the Wikidata resource *'Q7750866'*, which is 'The Medical Journal of Australia', instead of the resource *'Q1517084'*, which is the film 'The Medic'. The reason is probably that the properties of the resource *'Q7750866'* are more relevant to the question (in accordance with the relevance score described in section 1) than those of the resource *'Q1517084'*. For example, the property 'language of work or name'(P407) of the resource associated with the journal is more relevant to the question than the property 'original language of film or TV show'(P364) of the resource associated with the film. However, the method correctly recognizes the resource '*Medic_(TV_series)*' in DBpedia, which has the property 'language'(dbp).

4.3.3. Answer Composition

The performance of MuHeQA for generating answers to single fact questions by obtaining information from multiple Knowledge Graphs was measured. On this occasion the SimpleQuestions dataset

[18] https://github.com/egerber/spaCy-entity-linker
[19] https://www.dbpedia-spotlight.org

was used on both knowledge graphs, Wikidata and DBpedia, to compare the performance with other state-of-the-art approaches such as STaG-QA [11], SYGMA [8] and Falcon 2.0 [7]. While Falcon 2.0 is not a KGQA system itself, it allows generating the SPARQL query based on the entities and relations it identifies [11]. Due to differences in the conceptual organization of the knowledge graphs behind the SimpleQuestions dataset, the directionality of equivalent predicates in Freebase (i.e. original) and DBpedia or Wikidata may differ. For example, the DBpedia predicate *dbo:birthPlace* takes a person as the subject and a location as the object, whereas the equivalent predicate in Freebase *fb:location/location/people born here* inverts the subject and object [30]. Table 4 shows the results considering only questions whose directionality holds between the Knowledge Graphs (i.e. 'forward' type questions).

Table 4
Performance based on Knowledge Graph-oriented QA

System	*precision*	*recall*	*f1*
MuHeQA	59.70	56.33	57.97
Falcon 2.0 [7]	34.00	41.10	36.30
STaG-QA$_{pre}$ [11]	**60.2**	**63.2**	**61.7**
SYGMA [8]	42.00	55.00	44.00

The results show that our approach offers a performance close to the best system, STaG-QA, and better than other approaches specific to KGQA. However, one of the weak points is the *recall*, which means that our approach has to improve in response elaboration. The answer is perhaps too straightforward, and we should be concerned with constructing more complex responses.

4.3.4. Unstructured Data Sources

Since MuHeQA also supports unstructured knowledge sources, its performance was evaluated on QA pairs based on text documents. The answers are composed from the set of passages that are considered relevant to a given question. In this scenario, while extractive QA-based approaches, such as ours, highlight the span of text that answers a query, generative QA-based approaches create answers based on pieces of text they consider relevant. Retrieval-Augmented Generation models (RAGs) are based on generative QA techniques and they have recently attracted the attention of researchers due to their high performance in QA tasks [35]. They accommodate fine-tuned language models in modular pipelines [36] [37] or end-to-end architectures [13] to retrieve textual passages that are used to create the answers from a given question. Both approaches were compared to answer the questions provided on three domain-specific datasets. The COVID-19 dataset contains 1,765 QA pairs created from 5,000 full-text scientific articles selected from the CORD-19 corpus [38]. The News dataset contains 5,126 human annotated QA pairs based on 10,000 news articles selected from the NewsQA dataset [39]. And the Conversation dataset contains 3,260 QA pairs based on 10,000 conversations retrieved from the QAConv dataset [40]. Exact Match (EM) and F1 score were used as evaluation metrics. The EM score computes the word level exact match between the predicted answer and the real answer. The F1-score calculates the number of words in the predicted answer that are aligned with the real answer regardless of the order. The results are showed in Table 5.

The results show a similar behaviour to the evaluation based on knowledge graphs and, in general, offer high performance. The answers created by our algorithm are not as elaborate as those in the evaluation dataset, which were created manually, and this penalises the performance of our system. For example, given the question *"How many children were infected by HIV-1 in 2008-2009, worldwide?"*, the answer inferred by our system is *"more than 400,000"*, while the correct answer is *"more than 400,000 children were infected worldwide, mostly through MTCT and 90% of them lived*

Table 5
Performance based on Document-oriented QA

System	CovidQA		NewsQA		ConversationQA	
	EM	*f1*	*EM*	*f1*	*EM*	*f1*
MuHeQA	8.15	**22.24**	12.38	16.85	**29.96**	37.79
RAG-end2end [13]	**8.32**	19.51	**14.08**	**23.70**	25.95	**37.96**

in sub-Saharan Africa". This behaviour has already been shown previously when evaluating performance with graph-based knowledge bases and is one of the current limitations of our proposal. The creation of more elaborated responses is one of the future lines of work that we need to address. On the other hand, the accuracy is quite high, offering a promising overall performance. This makes us optimistic not only for the handling of documentary knowledge bases, but also for producing higher quality summaries from KGs that will allow us to achieve a performance like the one obtained for documentary datasets.

5. Conclusion

In this paper, we have presented the MuHeQA system that provides QA capabilities over multiple and heterogeneous knowledge graphs. Both qualities, i.e. that it supports one or several Knowledge Graphs and that they can have different schemas or formats, even being unstructured data sources, are achieved because it does not require building formal queries from the NL query. We introduce a new way of querying Knowledge Graphs based on textual summaries created from resource properties, instead of SPARQL queries. We propose several mechanisms to increase coverage, by recognizing entities and key concepts in queries, as well as discovering associated resources in the Knowledge Graph. The performance of MuHeQA has been evaluated both on knowledge graphs, such as Wikidata and DBPedia, and on documentary databases, such as Covid-19 QA, and offers close to state-of-the-art behavior without the need to train supervised models that require domain-specific data. We are optimistic in the capability that this approach offers and our next steps are to support multi-hop queries to accept complex questions and to elaborate richer answers.

Acknowledgments

This work is supported by the DRUGS4COVID++ project, funded by Ayudas Fundación BBVA a equipos de investigación científica SARS-CoV-2 y COVID-19.

References

[1] A. Pereira, A. Trifan, R.P. Lopes and J.L. Oliveira, Systematic review of question answering over knowledge bases, *IET Software* (2022), 1–13, doi:10.1049/sfw2.12028.

[2] H. Sun, T. Bedrax-Weiss and W.W. 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 (EMNLP-IJCNLP)*, 2019, pp. 2380–2390, doi:10.18653/v1/D19-1242.

[3] W. Xiong, X. Li, S. Iyer, J. Du, P. Lewis, W.Y. Wang, Y. Mehdad, S. Yih, S. Riedel, D. Kiela and B. Oguz, Answering Complex Open-Domain Questions with Multi-Hop Dense Retrieval, 2021, doi:10.48550/arXiv.2009.12756.

[4] H. Sun, W.W. Cohen and R. Salakhutdinov, Iterative Hierarchical Attention for Answering Complex Questions over Long Documents, 2021, doi:10.48550/arXiv.2106.00200.

[5] M. Yani and A.A. Krisnadhi, Challenges, Techniques, and Trends of Simple Knowledge Graph Question Answering: A Survey, *Information* (2021), doi:10.3390/info12070271.

[6] Y. Lan, G. He, J. Jiang, J. Jiang, W.X. Zhao and J.-R. Wen, A Survey on Complex Knowledge Base Question Answering: Methods, Challenges and Solutions, in: *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, International Joint Conferences on Artificial Intelligence Organization, 2021, pp. 4483–4491, doi:10.24963/ijcai.2021/611.

[7] A. Sakor, K. Singh, A. Patel and M.-E. Vidal, Falcon 2.0: An Entity and Relation Linking Tool over Wikidata, Association for Computing Machinery, 2020, pp. 3141–3148, doi:10.1145/3340531.3412777.

[8] S. Neelam, U. Sharma, H. Karanam, S. Ikbal, P. Kapanipathi, I. Abdelaziz, N. Mihindukulasooriya, Y.-S. Lee, S. Srivastava, C. Pendus, S. Dana, D. Garg, A. Fokoue, G.P.S. Bhargav, D. Khandelwal, S. Ravishankar, S. Gurajada, M. Chang, R. Uceda-Sosa, S. Roukos, A. Gray, G. Lima, R. Riegel, F. Luus and L.V. Subramaniam, SYGMA: A System for Generalizable and Modular Question Answering Over Knowledge Bases, in: *Findings of the Association for Computational Linguistics: EMNLP 2022*, Association for Computational Linguistics, 2022, pp. 3866–3879.

[9] A. Abujabal, M. Yahya, M. Riedewald and G. Weikum, Automated Template Generation for Question Answering over Knowledge Graphs, in: *Proceedings of the 26th International Conference on World Wide Web*, International World Wide Web Conferences Steering Committee, 2017, pp. 1191–1200, doi:10.1145/3038912.3052583.

[10] Y. Lan and J. Jiang, Query Graph Generation for Answering Multi-hop Complex Questions from Knowledge Bases, in: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Association for Computational Linguistics, 2020, pp. 969–974, doi:10.18653/v1/2020.acl-main.91.

[11] S. Ravishankar, D. Thai, I. Abdelaziz, N. Mihindukulasooriya, T. Naseem, P. Kapanipathi, G. Rossiello and A. Fokoue, A Two-Stage Approach towards Generalization in Knowledge Base Question Answering, in: *Findings of the Association for Computational Linguistics: EMNLP 2022*, Association for Computational Linguistics, 2022, pp. 5571–5580.

[12] S. Pramanik, J. Alabi, R.S. Roy and G. Weikum, UNIQORN: Unified Question Answering over RDF Knowledge Graphs and Natural Language Text, *Computing Research Repository* (2021).

[13] S. Siriwardhana, R. Weerasekera, E. Wen, T. Kaluarachchi, R. Rana and S. Nanayakkara, Improving the Domain Adaptation of Retrieval Augmented Generation (RAG) Models for Open Domain Question Answering, *Transactions of the Association for Computational Linguistics* (2023), 1–17.

[14] A. Akbik, D. Blythe and R. Vollgraf, Contextual String Embeddings for Sequence Labeling, in: *COLING 2018, 27th International Conference on Computational Linguistics*, Association for Computational Linguistics, 2018, pp. 1638–1649.

[15] O. Agarwal, H. Ge, S. Shakeri and R. Al-Rfou, Knowledge Graph Based Synthetic Corpus Generation for Knowledge-Enhanced Language Model Pre-training, 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. 3554–3565, doi:10.18653/v1/2021.naacl-main.278.

[16] Y. Zhu, J. Wan, Z. Zhou, L. Chen, L. Qiu, W. Zhang, X. Jiang and Y. Yu, Triple-to-Text: Converting RDF Triples into High-Quality Natural Languages via Optimizing an Inverse KL Divergence, in: *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR'19, Association for Computing Machinery, 2019, pp. 455–464–, doi:10.1145/3331184.3331232.

[17] B. Oguz, X. Chen, V. Karpukhin, S. Peshterliev, D. Okhonko, M. Schlichtkrull, S. Gupta, Y. Mehdad and S. Yih, UniK-QA: Unified Representations of Structured and Unstructured Knowledge for Open-Domain Question Answering, in: *Findings of the Association for Computational Linguistics: NAACL 2022*, Association for Computational Linguistics, 2022, pp. 1535–1546, doi:10.18653/v1/2022.findings-naacl.115.

[18] P. Lewis, L. Denoyer and S. Riedel, Unsupervised Question Answering by Cloze Translation, in: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, Association for Computational Linguistics, 2019, pp. 4896–4910, doi:10.18653/v1/P19-1484.

[19] Y. Zhou, X. Geng, T. Shen, W. Zhang and D. Jiang, Improving Zero-Shot Cross-lingual Transfer for Multilingual Question Answering over Knowledge Graph, 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. 5822–5834, doi:10.18653/v1/2021.naacl-main.465.

[20] J. Devlin, M.-W. Chang, K. Lee and K. 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, Volume 1 (Long and Short Papers)*, Association for Computational Linguistics, 2019, pp. 4171–4186, doi:10.18653/v1/N19-1423.

[21] P. Rajpurkar, J. Zhang, K. Lopyrev and P. Liang, SQuAD: 100,000+ Questions for Machine Comprehension of Text, in: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, 2016, pp. 2383–2392, doi:10.18653/v1/D16-1264.

[22] S. Raza, B. Schwartz and L.C. Rosella, CoQUAD: a COVID-19 question answering dataset system, *BMC Bioinformatics* (2022), doi:10.1186/s12859-022-04751-6.

[23] C. Nikas, P. Fafalios and Y. Tzitzikas, Two-stage Semantic Answer Type Prediction for Question Answering using BERT and Class-Specificity Rewarding, in: *Proceedings of the SeMantic AnsweR Type prediction task (SMART) at ISWC 2020 Semantic Web Challenge co-located with the 19th International Semantic Web Conference (ISWC 2020), Virtual Conference, November 5th, 2020*, CEUR Workshop Proceedings, CEUR-WS.org, 2020, pp. 19–28.

[24] C. Badenes-Olmedo, D. Chaves-Fraga, M. Poveda-VillalÓn, A. Iglesias-Molina, P. Calleja, S. Bernardos, P. MartÍn-Chozas, A. Fernández-Izquierdo, E. Amador-Domínguez, P. Espinoza-Arias, L. Pozo, E. Ruckhaus, E. González-Guardia, R. Cedazo, B. López-Centeno and O. Corcho, Drugs4Covid: Drug-driven Knowledge Exploitation based on Scientific Publications, 2020, doi:10.48550/arXiv.2012.01953.

[25] J. Berant, A. Chou, R. Frostig and P. Liang, Semantic Parsing on Freebase from Question-Answer Pairs, in: *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, 2013, pp. 1533–1544.

[26] 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: *The Semantic Web – ISWC 2019*, Springer International Publishing, 2019, pp. 69–78, isbn:978-3-030-30796-7.

[27] E. Kacupaj, H. Zafar, J. Lehmann and M. Maleshkova, VQuAnDa: Verbalization QUestion ANswering DAtaset, in: *The Semantic Web*, Springer International Publishing, 2020, pp. 531–547, doi:10.1007/978-3-030-49461-2_31.

[28] A. Bordes, N. Usunier, S. Chopra and J. Weston, Large-scale Simple Question Answering with Memory Networks, *Computing Research Repository* (2015).

[29] D. Diefenbach, T.P. Tanon, K. Singh and P. Maret, Question Answering Benchmarks for Wikidata, in: *Proceedings of the ISWC 2017 Posters Demonstrations and Industry Tracks co-located with 16th International Semantic Web Conference*, 2017.

[30] M. Azmy, P. Shi, J. Lin and I. Ilyas, Farewell Freebase: Migrating the SimpleQuestions Dataset to DBpedia, in: *Proceedings of the 27th International Conference on Computational Linguistics*, Association for Computational Linguistics, 2018, pp. 2093–2103.

[31] S. Schweter and A. Akbik, FLERT: Document-Level Features for Named Entity Recognition, 2021, doi:10.48550/arXiv.2011.06993.

[32] L. Zhuang, L. Wayne, S. Ya and Z. Jun, A Robustly Optimized BERT Pre-training Approach with Post-training, in: *Proceedings of the 20th Chinese National Conference on Computational Linguistics*, Chinese Information Processing Society of China, 2021, pp. 1218–1227, doi:10.1007/978-3-030-84186-7$_3$1.

[33] D. Diomedi and A. Hogan, Question Answering over Knowledge Graphs with Neural Machine Translation and Entity Linking, 2021, doi:10.48550/arXiv.2107.02865.

[34] P.N. Mendes, M. Jakob, A. García-Silva and C. Bizer, DBpedia Spotlight: Shedding Light on the Web of Documents, in: *Proceedings of the 7th International Conference on Semantic Systems*, I-Semantics '11, Association for Computing Machinery, 2011, pp. 1–8–, doi:10.1145/2063518.2063519. ISBN 9781450306218.

[35] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, S. Riedel and D. Kiela, Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks, in: *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS'20, Curran Associates Inc., 2020, isbn:9781713829546.

[36] V. Karpukhin, B. Oguz, S. Min, P. Lewis, L. Wu, S. Edunov, D. Chen and W.-t. Yih, Dense Passage Retrieval for Open-Domain Question Answering, in: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Association for Computational Linguistics, 2020, pp. 6769–6781, doi:10.18653/v1/2020.emnlp-main.550.

[37] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov and L. Zettlemoyer, BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension, in: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Association for Computational Linguistics, 2020, pp. 7871–7880, doi:10.18653/v1/2020.acl-main.703.

[38] L.L. Wang, K. Lo, Y. Chandrasekhar, R. Reas, J. Yang, D. Burdick, D. Eide, K. Funk, Y. Katsis, R.M. Kinney, Y. Li, Z. Liu, W. Merrill, P. Mooney, D.A. Murdick, D. Rishi, J. Sheehan, Z. Shen, B. Stilson, A.D. Wade, K. Wang, N.X.R. Wang, C. Wilhelm, B. Xie, D.M. Raymond, D.S. Weld, O. Etzioni and S. Kohlmeier, CORD-19: The COVID-19 Open Research Dataset, in: *Proceedings of the 1st Workshop on NLP for COVID-19 at ACL 2020*, Association for Computational Linguistics, 2020.

[39] A. Trischler, T. Wang, X. Yuan, J. Harris, A. Sordoni, P. Bachman and K. Suleman, NewsQA: A Machine Comprehension Dataset, in: *Proceedings of the 2nd Workshop on Representation Learning for NLP*, Association for Computational Linguistics, 2017, pp. 191–200, doi:10.18653/v1/W17-2623.

[40] C.-S. Wu, A. Madotto, W. Liu, P. Fung and C. Xiong, QAConv: Question Answering on Informative Conversations, in: *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Association for Computational Linguistics, 2022, pp. 5389–5411, doi:10.18653/v1/2022.acl-long.370.