Knowledge and Information Systems (2024) 66:453–479
https://doi.org/10.1007/s10115-023-01966-8

REGULAR PAPER

A template-based approach for question answering over knowledge bases

Anna Formica[1] · **Ida Mele**[1] · **Francesco Taglino**[1]

Received: 28 December 2022 / Revised: 27 April 2023 / Accepted: 8 August 2023 /
Published online: 2 September 2023

Abstract
In this paper, we address the problem of answering complex questions formulated by users in natural language. Since traditional information retrieval systems are not suitable for complex questions, these questions are usually run over knowledge bases, such as Wikidata or DBpedia. We propose a semi-automatic approach for transforming a natural language question into a SPARQL query that can be easily processed over a knowledge base. The approach applies classification techniques to associate a natural language question with a proper query template from a set of predefined templates. The nature of our approach is semi-automatic as the query templates are manually written by human assessors, who are the experts of the knowledge bases, whereas the classification and query processing steps are completely automatic. Our experiments on the large-scale CSQA dataset for question-answering corroborate the effectiveness of our approach.

Keywords Knowledge base · Question answering · Template-based question classification

1 Introduction

Question answering (Q/A) systems [1, 2] have recently received a lot of attention, and their success is due to the fact that they allow answering natural language (NL) questions posed by the users. Such answers are taken from a large data collection and, more specifically, from knowledge bases (KBs). Compared to traditional information retrieval (IR) systems, Q/A systems do not provide a list of results ranked by their relevance to the user's query but rather return one crisp answer that must satisfy the user's request. Hence, the user can save valuable time as he/she does not need to inspect a long list of ranked results. However, the task is more

✉ Ida Mele
ida.mele@iasi.cnr.it

Anna Formica
anna.formica@iasi.cnr.it

Francesco Taglino
francesco.taglino@iasi.cnr.it

[1] Istituto di Analisi dei Sistemi ed Informatica (IASI) "Antonio Ruberti", National Research Council, via dei Taurini, 19, 00185 Rome, Italy

difficult as the answer returned to the user's question must be relevant and accurate. For this reason, the KBs help the answer formulation consistently.

Q/A systems can be very different and used for several purposes, therefore they have different architectures. In any case, most of the systems are based on the general framework described by Voorhees [3], where the question is first analyzed to determine the possible answer's type, i.e., *question classification*. Then, document or passage retrieval is performed to get a pool of potential results. Named entities are extracted from the pool of candidate results, and only those that match the answer type (e.g., person, country, organization) are taken into account. This allows for filtering out all the results that cannot be consistent with the question posed by the user.

In Q/A scenarios, the questions can be broadly categorized as factoid and non-factoid. The former are questions requesting a fact where the answer is a simple word or more words often corresponding to a named entity (i.e., a person, an organization, or a location). The latter are questions where the answer is expected to be more descriptive and explanatory. Hence, question classification is an important component of any Q/A system as it allows the type of the answer to be determined. In addition, it has many applications, for example, it can be used in the *question triage process* to identify whether the question can be answered by an automatic system or whether it is better to ask a human. In this paper, we address Q/A for factoid questions that can be simple questions or more complex ones, e.g., involving logical union or intersection, and require some form of reasoning.

In this work, we address Q/A by following a *template-based* approach [4], where a template represents a type of query that once instantiated is run over a KB to get the result. An NL question is mapped to a proper query template, and for this purpose, we propose to use classification techniques, such as Gradient-Boosted Decision Trees (GBDT) [5], Naïve Bayes [6], and Support Vector Machines (SVM) [7], in order to find the most suitable query template from a set of predefined templates. To the best of our knowledge, this is the first work that applies classification techniques for mapping a user question to a query template, in this paper referred to as *template-based question classification*.

Other approaches propose to directly create the template from the NL questions [8] or to directly answer the user question [9]. Although these fully automatic systems scale well and do not require manual assessors for writing query templates, they suffer from low precision [9] or they need large training data with question-answer pairs which are costly to gather [8]. We aim at improving the trade-off between scalability and the quality of results. In real-world scenarios, the users are unsatisfied with incorrect answers, and they need to continuously rephrase the NL questions to get the correct results. We believe that the high precision of the results is mandatory to ensure high satisfaction of the end users. Our approach maps the NL question to a query template thanks to a classifier. Finding the best query template from a set of pre-defined query templates leads to higher performance, with an average precision of about 65.44% with respect to 10.52% achieved by [9] as shown in Sect. 5. Moreover, the classifier can handle new patterns as it correctly predicts the labels of unseen grammar patterns of the NL questions. In our proposed template-based question classification, the classifier leverages both *syntactic*, e.g., Part-of-Speech (PoS) tags, and *semantic* features, e.g., most discriminative words. In addition, this proposal overcomes the main drawback of the template-based approaches for which templates are query skeletons that closely correspond to the structures of the queries, leading therefore to an explosion of the number of possible templates [10].

Nowadays, most of the approaches for question classification are based on supervised machine-learning techniques leveraging semantic, syntactic, and lexical features (e.g., n-grams) [11]. While in Q/A system literature, the traditional problem is classifying questions

with the purpose of predicting the answer type (e.g., human, location) [12, 13], in our contribution, we aim at mapping a user's question to a query template. In other words, we address the problem of choosing the most suitable query template from a set of predefined templates as a classification task. Our approach is supervised as it needs labeled data to train the classifier, and we rely on the large-scale dataset named "Complex-Sequential Question Answering" (CSQA), presented in [9]. The dataset consists of 200K dialogues with a total of 1.6M turns (i.e., questions), and also includes answers taken from Wikidata. The questions are categorized on the basis of a taxonomy of question classes, in particular, logical, quantitative, and comparative. Among them, we kept out only the indirect and the incomplete questions that are properly related to conversational aspects [14], and go beyond the scope of this paper.

As mentioned above, completely automatic template-generation techniques lead to an explosion in the number of templates because the structure of the query template is similar to the structure of the NL question [10]. Moreover, as we will see in the Related Work section, they often need training data where the user questions are paired with their answers, and this is costly to achieve due to the labeling effort [8]. Our approach keeps the number of templates small, plus it does not need expensive question-answer pairs in the training data since it is based on classification for matching a question to its best query template. Similarly to our proposal, other works are based on a predefined set of hand-crafted templates. These sets are often small, the templates consider only some of the SPARQL operators [15], or their coverage is limited to the simple questions [16]. As we will see in Sect. 5, our approach covers many different types of questions from structurally simple to more complex ones such as logical, quantitative, or comparative questions.

Overall, the contribution of this paper is threefold: (i) a Q/A framework for answering complex questions expressed in NL relying on a template-based classification, (ii) the investigation of GBDT, Naïve Bayes, and SVM classification approaches for detecting the best query template, and (iii) a novel set of SPARQL query templates to run over a KB that can be used for very different types of NL questions.

The paper is organized as follows. The related work is presented in Sect. 2. In Sect. 3, the method is presented by introducing first the general architecture and, successively, our taxonomy of question classes. Section 4 illustrates the query templates, in particular, for logical, comparative, quantitative, and simple questions. In Sect. 5, the experiments are given, with the addressed dataset, and the evaluation of the results. Lastly, Sect. 6 concludes the paper.

2 Related work

In the literature, there are several approaches dealing with Q/A systems, see for instance [17] and [18] for *Community* Q/A platforms, or [19] that proposes a template-based question generation approach without relying on a KB. In this section, we focus on the methods underlying Q/A systems over KBs. In [4], such methods can be classified according to four different architectures: *semantic parsing pipelines*, *subgraph matching*, *template-based question answering over KB (template-based KBQA)*, *KBQA based on information extraction*.

The first category of architectures, the semantic parsing pipelines, relies on natural language processing (NLP) techniques, such as tokenization, Named Entity Recognition, PoS tagging, and Entity Linking, with the aim of generating the queries and the corresponding answers according to filters, in a straightforward way. Examples are the proposals presented

in [20–22]. However, as mentioned in [4], the approach of semantic parsing pipelines is the oldest one and seems to have reached its maturity.

According to the subgraph matching architecture, a query subgraph is built directly from the KB, see, for instance, the works presented in [23, 24]. Essentially, they are data-driven frameworks addressing the disambiguation and the query evaluation steps by relying on subgraph matching problems. In general, these approaches require costly tasks, for instance, graph similarity techniques, and therefore have to deal with scalability problems.

The third category of architecture is the template-based KBQA to which our proposal belongs. In [4], the authors claim that there are a few proposals in the literature about template-based methods, whereas studying this kind of architecture is very promising and worthy of investigation. Some of the template-based Q/A approaches rely on query skeletons, i.e., templates, having slots that have to be filled with the entities and the relations of the KB to be questioned. In general, they aim at transforming an NL question into a SPARQL query. For example, Bast and Haussmann [16] tackled the problem of automatic translation of simple NL questions to their matching SPARQL queries using three manually constructed query templates. The motivation behind their work is that even structurally simple questions (e.g., "*Who is the CEO of Apple?*") suffer from the entity-matching problem. This is due to the variety of natural language, therefore the question may contain variants of the names used in the KB (synonyms). Moreover, different entities may have the same name in the KB (polysemy). The authors proposed the *Aqqu* system that, given an NL question, generates different query candidates, then applies the learning-to-rank technique to rank the candidates and return the highest-ranking query. The three query templates cover questions with a simple structure, whereas our proposal addresses the challenging problem of complex questions that exhibit compositionality (e.g., questions have multiple clauses connected by logical operators). Our task is similar to the one addressed in [15]. However, differently from our work that uses Naïve Bayes, SVM, and GBDT classifiers, the mentioned paper applies a recursive neural network method for question classification, and it does not address queries involving `UNION`, `FILTER`, `MIN`, and `MAX` SPARQL operators; hence, it cannot answer many complex questions. In addition, as stated by the authors, the performance of their proposal varies considerably per dataset on the basis of the complexity and expressiveness of the questions. Other approaches are based on construction rules as [25]. The authors impose constraints that depend on the specific Knowledge Graph (KG), i.e., DBpedia, and assume that the KG is complete, i.e., exhaustive. As an example, they suppose that any location resource is represented as an instance of the *dbo:Place* class, ignoring the fact that, in reality, DBpedia may have incomplete information.

Other template-based approaches aim at automatically creating the templates from the NL questions. According to [10], the main drawback of these approaches is that often a template tightly corresponds to the structure of the question, with the consequent explosion in the number of templates. One of the most representative automatic approach for template generation is QUINT [8], and its evolution named Never Ending Question Answering (NEQA) [26]. In particular, QUINT is a system that automatically learns utterance-query templates solely from user questions paired with their answers. As shown by the authors in the experimental results, the proposed method achieves an average F-score of 49.2% on the ComplexQuestions dataset [8]. Despite this approach being able to answer compositional questions without having learned any templates for the entire question, it requires large training data with question–answer pairs and suffers from the lack of appropriate templates for some questions. NEQA [26] addresses these limits. with a continuous learning-based method. It attempts to answer a question using a previously learned template. In the case it fails, it tries to answer the question by relying on a similarity function against the set of already answered

questions. Also, Zheng et al. [27] address the limits of the automatic template-generation approaches with a system for understanding NL questions via binary templates. The approach learns the pairs of NL patterns and KG triples offline. Then, it decomposes the user question into constituent parts based on the pre-generated templates and gets a structured query to run over the KG. Compared to our proposal where we consider complex questions with logical, quantitative, and comparative operators, Zheng et al. handle simple and complex questions, where a question is defined as complex if it contains more than one fact connected by an "and", "or", "but" grammar conjunctions (e.g., *"Who is married to an actor and starred in Philadelphia?"*). Other approaches focus on a subset of questions with a specific type of answer. For instance, [28] investigates a model based on Neural Machine Translation and proposes *Neural SPARQL Machines* to learn pattern compositions. Although the approach is completely automatic, it is limited to the particular *dbo:Eukaryote* class of DBpedia.

The fourth category of architecture, i.e., the one based on information extraction, relies on machine-learning techniques and adopts methods for extracting triples directly from the KB. Luo et al. [29] present a BERT-based approach for single-relation Q/A, consisting of two models: entity linking and relation detection. Entity linking is performed through a pre-trained BERT model and a heuristic algorithm to reduce the noise in the candidate facts. For relation detection, a BERT-based model with relation-aware attention is proposed to preserve the original interactive information with the user. Although this approach achieves state-of-the-art accuracy, it was tested on a dataset made only of simple questions. Within this kind of architecture, it is worth recalling the system presented in [9], which has been proposed by the authors together with the CSQA dataset addressed in our experiment. It is a Q/A and a Dialog System combining a neural sequence-to-sequence model with a Key-Value Memory Network. According to the experimental results given in the paper, the performance of the proposed model achieves a 10.52% precision and a 27.22% recall for logical questions. Furthermore, in [30], neural network models are investigated in order to map natural language statements to SPARQL expressions. As stated by the authors, the performance of their proposal heavily depends on the selected dataset, the complexity of the questions, and the vocabulary size.

Another line of research tackles the problem of answering complex questions over KBs using Reinforcement Learning (RL-based approaches). These approaches have good performance but share common limitations due to the fact that the agent is usually misled by aimless exploration and sparse or delayed rewards that lead to a large number of spurious relation paths. To overcome these issues, Zhang et al. [31] instead of using a random-walk agent, propose a new Adaptive Reinforcement Learning (ARL) framework for generating candidate paths through the application of three atomic operations until the target entity is reached. Then, they proposed a semantic policy network taking into account character- and sentence-level semantics for choosing the optimal actions that guide the agent. Lastly, they introduced a new reward function that takes into account both the relation paths and the target entities for alleviating the issue of delayed and sparse rewards. The approach was tested on two Chinese benchmark datasets and three English datasets with different percentages of complex questions that require multi-hop reasoning to be solved.

Along different lines of research, [32] aims at answering a given question by optimizing the ranked lists of SPARQL query candidates produced by knowledge graph Q/A systems. In [33], the authors propose a method to generate questions directly from SPARQL, in the absence of a sufficient number of resources (low-resources), which is a topic non-adequately investigated in the literature. Lastly, Visual Question Answering (VQA) is an orthogonal line of research dealing with vision-and-language tasks. Yu et al. experimented with the application of deep modular co-attention networks [34] and multimodal attention networks [35] for improving the understanding of both visual and textual contents.

As mentioned in the Introduction, we propose a semi-automatic template-based approach for transforming an NL question into a SPARQL query. In particular, this is a novel approach that relies on classification techniques for mapping a user question to the most suitable query template from a set of predefined templates. With regard to the main drawback of these approaches related to the strict correspondence of a question with a template, in our proposal, a template corresponds to a set of questions (a class), where each class gathers different PoS patterns with similar structures. In addition, in our approach, the question classes have been defined by further refining the types of questions identified in [9], as described in Sect. 5.1.

2.1 Question classification

Question classification represents an important component of any Q/A system. It aims at assigning a label to a question based on the type of the expected answer. For example, given the question "Who was the first Prime Minister of Canada?" the answer would be a person [12]. The question classes are usually defined upfront and represent a question taxonomy. These approaches are orthogonal to our work as they aim at predicting the type of the answer, whereas we classify questions with the purpose of finding the best query template.

Q/A systems have received a lot of attention in these years from researchers. As a matter of fact, NIST Text REtrieval Conference (TREC)[1] has organized several tracks devoted to the most important challenges in Q/A systems. TREC QA track is a large-scale Q/A evaluation that started in 1999 with TREC-8 [36], and each year has been proposing new questions and data collections to foster research in the field of Q/A systems. Although the question-classification task is popular and valuable for Q/A systems, there is not a unique taxonomy, as many authors proposed their own taxonomy on the basis of their research goal. Taxonomies can be flat (one-level classes) or hierarchical (multi-level classes). UIUC is a very popular taxonomy proposed by Li and Roth [37] and adopted by several authors in their research. It is a hierarchical taxonomy consisting of 6 coarse-grained classes (abbreviation, entity, description, human, location, numeric), and 50 fine-grained classes that are subclasses of the 6 more general ones (e.g., city, country, and mountain belong to location). Compared to flat taxonomies, this hierarchical taxonomy provides more flexibility, allowing for example the classification of questions at different levels of granularity or leveraging the hierarchical structure to improve the model learning.

Question classification can be performed in several ways, according to rule-based or machine-learning approaches. Rule-based question classification relies on hand-crafted rules that are predefined and based on the question taxonomy [38]. These classifiers have the advantage to be extremely powerful and achieve good precision in predicting the question classes. On the other hand, defining rules is very costly both in terms of time and human effort. Moreover, heuristic rules tend to be very specific and tailored for one type of question taxonomy; therefore, they cannot be easily generalized, and their application to different taxonomies or domains requires a laborious effort. One of the main limitations of rule-based question classifiers is that they cannot scale to a large number of questions or syntactical structures. To overcome this, statistical question classification approaches based on machine learning have been proposed. These approaches are very popular and can be unsupervised or supervised. Unsupervised learning is more challenging as it aims at training the machines to do something without providing labeled data. Examples of unsupervised learning approaches are reinforcement learning and clustering. On the other hand, supervised learning is based on providing some known examples to the machine (training set). Hence, the machine can be trained on

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

such data in order to learn a model and predict the outcomes for new unknown examples (test set). In particular, a supervised classifier can be trained on a set of questions for which we know the corresponding classes, and then it predicts the classes of the questions belonging to the test data. Most of them are based on well-known classification approaches (e.g., SVM, Naïve Bayes, decision trees) and rely on syntactic, semantic, and linguistic features extracted from the questions. IBM TREC-9 system [39] applies maximum-entropy models and leverages both syntactic and semantic features to label the questions according to the MUC categories proposed by [40]. Li and Roth [37] also use the syntactic and semantic features but their approach is based on The SNoW architecture. Moreover, for their experiments, they provided a superset of the TREC QA track dataset and proposed the above-mentioned UIUC taxonomy.

Zhang and Lee [13] present an SVM approach using only syntactic features. Their experimental results corroborate that SVM can get an accuracy of 90% on the coarse-grained level of classes proposed by [37]. Similarly, Metzler and Croft [12] proposed a question classification approach based on SVMs. Compared to [13], the authors trained different classifiers, one for each question word (e.g., who, what, when, where, why, and how). They also performed extensive experiments on three different datasets (TREC QA, UIUC, and MadSci) using both semantic and syntactic features. Their experimental results show that the proposed approach is robust and achieves good performance in different domains with just a little hand tuning (if needed).

Despite syntactic and semantic features allowing good results, some authors have experimented with lexical features, too. Mishra et al. [11] proposed an approach for question classification leveraging semantic, syntactic, and lexical features such as *n-grams*.

Although machine-learning approaches have better results than rule-based ones and can be easily adapted to different domains and taxonomies, the new trend is to combine machine learning with rule-based techniques. This is also due to the fact that understanding complex question patterns is not easy at all, and only statistical techniques despite being very powerful cannot be sufficient, and some hand-crafted rules can help to improve the accuracy of the classifiers. Other authors proposed techniques that combine rule-based approaches and statistical machine learning. In [41], the authors described the Javelin system that applies both automatic learning parsing plus hand-crafted rules to improve performance.

Our task is similar to the above-mentioned approaches as we aim at mapping a user's question to a class. However, it differs from traditional question classification since it does not classify questions based on the answer type (e.g., person, country), rather it maps each question to its best query template to run over a KB.

3 The proposed method

In this section, we introduce the general architecture of our proposal followed by the taxonomy of question classes.

3.1 General architecture

We describe the general architecture of our framework, which is shown in Fig. 1. Given a question, the approach extracts the semantic and syntactic features (Step 1) and applies a template-based question classification to assign the question to a class in order to select the corresponding query template (Step 2). Indeed, each class is associated with a query

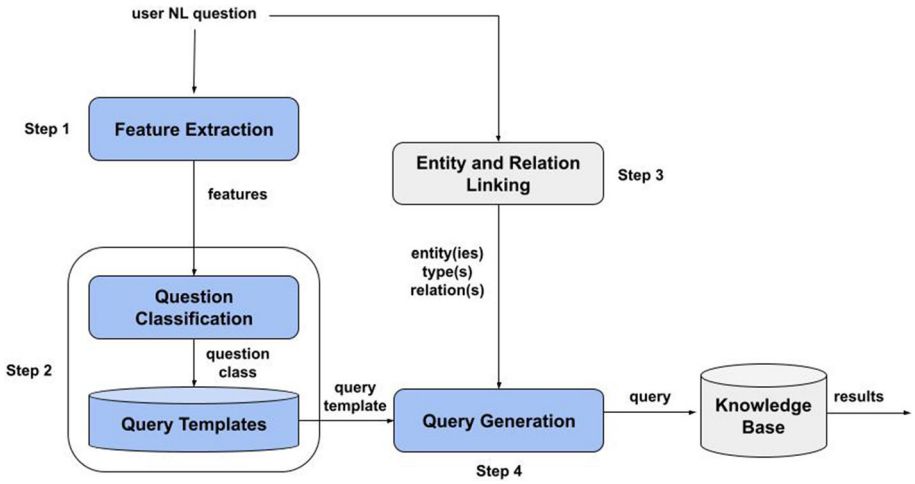


Fig. 1 Our proposed framework: from the user question to the SPARQL query's result. We highlighted the modules that represent our main contributions with the blue color. It is worth noting that the resulting query, generated by Step 4, can be used to query any knowledge base which is not pre-processed, such as DBpedia or Wikidata

template from a set of predefined ones. In parallel, the NL question is also the input of a module for Named-Entity Recognition (NER) and relation linking, that is the association of fragments of the text with resources in the KB, e.g., DBpedia or Wikidata (Step 3). Finally, such resources are used to instantiate the selected query template in order to formulate the actual SPARQL query that can be run over the KB (Step 4). We now describe in detail the steps of our approach.

- *Feature extraction (Step 1)* When the NL question is issued by the user, this module applies different techniques for feature extraction. In our task, we focus on syntactic and semantic features. Syntactic features are the PoS tags extracted from the question. In particular, we used the spaCy[2] tool for PoS tagging as it gives good results in terms of detecting the grammatical structure in NL sentences. As an example, given the NL question "Which watercourses flow through Hungary and Austria?", spaCy's PoS tags are "DET NOUN VERB ADP PROPN CCONJ PROPN PUNCT", where DET stands for *determiner*, NOUN for a *general noun*, VERB for *verb*, ADP for *adposition*, PROPN for a *proper noun* (e.g., a person, country, or organization that are usually recognized as a named entity), CCONJ for *coordinating conjunction*, and PUNCT for *punctuation* (e.g., question mark).
  In order to better discriminate different question classes, we also consider some semantic features (e.g., keywords extracted from the question). With feature selection, we could observe that the most important features are common terms, such as conjunctions (*and, or*) as they discriminate between union and intersection question classes. Also, common adjectives (*more*, *less*, or *approximately*) are important to discriminate among different sub-classes of the comparative and quantitative questions (see Sect. 3.2).
- *Template-based question classification (Step 2)* Given the features (PoS tags and discriminative keywords), the template-based question classification module first needs to create a feature matrix, then it applies a classifier to determine the best template for each NL

[2] https://spaCy.io/.

question. For creating the feature matrix, we rely on a Bag-of-Word model (BoW) [42, 43], which is a simplified representation of text, commonly used in NLP and IR. In BoW, the text (e.g., a document or a sentence) is represented as an unordered set of words and their frequencies (counts), hence the model associates more weight with those words that occur more frequently. The result is a matrix where rows represent documents/sentences (in our case NL questions posed by the user) and the columns are unique words (in our case the features, i.e., PoS tags and discriminative words). Each element in the matrix is the number of times the word/feature appears in the document/question, and it is equal to 0 if it does not appear. For our implementation, we applied the CountVectorizer[3] function that provides text pre-processing (e.g., stopwords removal), and we chose as input sequences of words up to 2 terms (i.e., bi-grams). The resulting feature matrix is passed to a classifier to determine for each question its most suitable class. Such a predicted class is used to retrieve the query template from the set of predefined templates. We experimented with different types of classifiers: Naïve Bayes [6], Support Vector Machines (SVM) [7] and Gradient-Boosted Decision Trees (GBDT) [5] that are described in Sect. 5.

- *Named-entity recognition and relation linking (Step 3)* This step is in charge to associate fragments of the NL questions with resources in the KB. For instance, if we consider again the question "Which watercourses flow through Hungary and Austria?", *watercourses*, *flow through*, *Hungary*, and *Austria* are associated with resources having specific identifiers defined in the chosen KB. This step employs tools for name-entity recognition and relation linking, such as Falcon,[4] TagMe,[5] or spaCy NER.[6]
  In particular, we investigated the spaCy and Falcon tools. The spaCy tool provides a fast statistical entity-recognition system able to assign labels to contiguous spans of tokens. Its default trained pipeline can identify a variety of named entities, such as people, companies, locations, organizations, and products, but it lacks a relation recognition system and, for this reason, we relied on Falcon, which can identify both named entities and relations, and returns the link to a DBpedia entry. Mapping the DBpedia entries to Wikidata is not always possible since some direct links between the two KBs are missing. The named-entity recognition and relation-linking problems are still very challenging and prone to many mistakes. Such errors affect the end-to-end results and do not allow us to understand the quality of the classification. Since we focus on the performance of the mapping between the NL question and the query template, rather than the ability of the NER tool to recognize named entities and relations, we used the named entities and relations from the original dataset which lead to more reliable results.
- *SPARQL query generation (Step 4)* This step instantiates the query template, identified in Step 2, by using the resources identified in Step 3. According to our example, given the resources extracted from the KB corresponding to *Hungary*, *Austria*, *watercourses*, and *flow through*, this step uses the query template identified by the question classifier for creating the SPARQL query to run over the KB.

[3] CountVectorizer is a function of the Skikit-learn package (https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html).

[4] https://labs.tib.eu/falcon/.

[5] https://tagme.d4science.org/tagme/.

[6] https://spaCy.io/api/entityrecognizer.

3.2 Taxonomy of question classes

Let us assume that the KB is a knowledge graph, such as Wikidata or DBpedia, whose pieces of knowledge are represented as triples of the form ⟨s,p,o⟩, where s and o are the *subject* and the *object* of the triple, respectively, and p is a *predicate* which expresses an ordered relation between them. For example, the triples asserting that the Danube flows through Hungary, or Hungary is traversed by the Danube, can be expressed, respectively, as follows:

⟨Danube, flow_through, Hungary⟩
⟨Hungary, traversed_by, Danube⟩

Furthermore, the entities (subject and object) of a triple can be associated with an *entity type*. For instance, according to Wikidata, the types of the entities Hungary and Danube are country, and river, respectively.

For our task, we relied on the CSQA dataset provided by Saha et al. in [9]. Some examples of questions from the dataset are reported in Table 1. We focused in particular on Single Relation and Single Entity type questions (see Sect. 5.1 for details). These types of questions may involve some form of reasoning among the triples of the knowledge graph. In particular, Single Relations are questions requiring logical inferences over the triples and include one of the logical operators AND, OR, NOT, which correspond to the logical intersection, union, and difference classes of questions, respectively. For instance, in Table 1, "Which film awards were won by Liv Ullmann or Lakshmi?" requires the union of the sets of the film awards that are subjects of the following triples:

⟨film_awards, won_by, Liv_Ullmann⟩
⟨film_awards, won_by, Lakshmi⟩

Single Entity types are complex questions that can be Comparative or Quantitative. The former needs a comparison between entities (Comparative More, Less, Approximately), and, after comparison, they can also require counting (Comparative Count over More, Less, Approximately). For example, in Table 1 the question "Which ship types have a greater number of administrative territories as registry ports than SS Coccolita?" is a Comparative More question asking for the list of ship types whose administrative registry ports are more than the ones of the *SS Coccolita* ship. Whereas "How many ship types have more number of administrative territories as registry port than Nymph?" is a Comparative Count over More question requiring first the comparison of the number of registry ports of a ship type with respect to the ones of the general cargo *Nymph* and, successively, the count of the ship types with more number of such ports with respect to the ones of *Nymph* (and the answer is such a number).

The latter, the Quantitative questions, involve standard aggregation functions such as Atmost, Atleast, Approximately, Equal, Max, and Min, analogously to the Comparative questions, they can also involve counting (Quantitative Count, Count over Atmost, Atleast, Approximately, Equal). For example, in Table 1, "Which musical instruments are played by at least 3683 people?" is a Quantitative Atleast question asking for the names of the musical instruments played by at least a given number of people, whereas "How many films did at least 9 people do the dubbing for?" is a Quantitative Count over Atleast question whose expected answer is an integer corresponding to the number of films at least a given number of people did the dubbing for.

Besides complex questions, we also addressed Simple Questions. These are essentially subject-, object-, or predicate-based questions, in the sense that the expected answers are the subjects, the objects, or the predicates of the triples, respectively, without requiring any

reasoning about them. An example of a simple subject-based question is given in Table 1, asking for the place where Carlos Recalde was born. Simple Questions have been extensively investigated in the literature and, of course, are easier to be answered [9].

Note that each of the CSQA classes of questions described above is organized according to Logical, Comparative, Quantitative, and Simple Questions. However, some classes include questions that may correspond to different query templates and, for this reason, in our approach, we divided them into sub-classes. For instance, the Comparative questions may contain *more* or *less* that correspond to two distinct query templates. After this refinement, we obtained a total of 21 classes, each one corresponding to a different query template. In Table 1, these question classes are shown, and for each class, we also provide an example taken from the dataset used in our experiment.

4 Query templates

This section introduces the SPARQL query templates that have been defined in order to address the questions' classes in Table 1. Successively, Tables 2, 3, 4, and 5 are shown, each presenting a group of templates. In such tables, the first column shows the SPARQL template where the query parameters are in bold, in the second one an example of a question is given, and the third column contains the fragments of the question that allow query parameters to be instantiated. In particular, these are the pieces of text that the *Entity and Relation linking* (see Fig. 1) module associates with resources in the knowledge graph.

In the templates, the `type_of` and `isa` predicates have been used to represent in a generic way the *instance of* and the *subclass of* relationships, respectively. In fact, different KBs can implement these relationships in different ways. For instance, in DBpedia they are represented by the *rdf:type* and the *rdfs:subClassOf* predicates, respectively, whereas, in Wikidata, they are represented by the *Property:P31* and *Property:P279* predicates, respectively. Depending on the underlying KB, when a template is instantiated the proper predicates are applied accordingly.

Furthermore, `nnx`, `nnpx`, and `propx`, where `x` is a natural number, are query parameters that represent resources in the knowledge graph identifying entity types, named entities, and predicates, respectively. In order to execute the actual query, the parameters need to be substituted by the ids of resources in the knowledge graph.

It is worth remembering that, according to the SPARQL syntax, the question mark (`?`) is used to introduce a variable, whereas the asterisk (`*`) after a predicate indicates the transitive closure of the relationship represented by that predicate. This means that for instance in the case of the triple `?x12 isa* nn1`, the resources identified by the variable `?x12` and the parameter `nn1` must be the first and the last nodes, respectively, of a path of any length involving only the `isa` predicate.

Templates for logical questions Table 2 shows the SPARQL templates that have been defined in order to address Logical questions, i.e., Logical Union, Logical Intersection, and Logical Difference questions. These templates are characterized by the `UNION` operator, the conjunction of all the search criteria (the conjunction is expressed through the symbol "."), and the `FILTER NOT EXISTS` operator that filters out the results adhering to specific criteria, respectively.

Table 1 Taxonomy of question classes with examples from CSQA

Class name	Example
Logical (Log.)	
Log. Union	Which film awards were won by Liv Ullmann or Lakshmi?
Log. Intersection	Which occupations are the job of Matei Vişniec and David D. Friedman?
Log. Difference	Which administrative territories are located in Eurasia but not Europe?
Comparative (Comp.)	
Comp. Count over More	How many ship types have more number of administrative territories as registry port than Nymph?
Comp. Count over Less	How many musical instruments are played by less number of people than harmonium?
Comp. Count over Approx	How many languages are around the same number of literary works in as Old Norse language?
Comp. More	Which ship types have greater number of administrative territories as registry port than SS Coccolita?
Comp. Less	Which works are recounted in lesser number of works of art than Pirates of the Caribbean?
Comp. Approx	Which land forms come from approximately the same number of ship types as Clotho Tessera?
Quantitative (Quant.)	
Quant. Count	How many administrative territories have a shared border with Spain?
Quant. Count over Atmost	How many works did at most 14 people do the dubbing for?
Quant. Count over Atleast	How many films did at least 9 people do the dubbing for?
Quant. Count over Approx	How many literary works illustrate around 1 work fictional universe?
Quant. Count over Equal	How many states have their official language as exactly 1 language?
Quant. Atmost	Which musical instruments can at most 7388 people perform with?
Quant. Atleast	Which musical instruments are played by at least 3,683 people?
Quant. Approx	Which ship types have approximately 1 administrative territory as their registry harbour?
Quant. Equal	Which land forms are exactly 2 bodies of water located on?
Quant. Max	Which administrative territories do max number of disasters finally stop at?
Quant. Min	Which geographic regions are min number of concepts located on?
Simple questions	
Simple question	Where was Carlos Recalde born?

Table 2 SPARQL templates for logical questions

SPARQL template	Example of question	Query parameters
Template for logical union questions		
`SELECT DISTINCT ?x11 ?x21 WHERE {{nnp1 prop1 ?x11 . ?x11 type_of ?x12 . ?x12 isa* nn1} UNION {nnp2 prop1 ?x21 . ?x21 type_of ?x22 . ?x22 isa* nn1}}`	Which countries do the Danube or the Rhine traverse?	`nn1`: countries `nnp1`: Danube `nnp2`: Rhine `prop1`: traverse
Template for logical intersection questions		
`SELECT DISTINCT ?x1 WHERE {nnp1 prop1 ?x1 . nnp2 prop1 ?x1 . ?x1 type_of ?x2 . ?x2 isa* nn1}`	Which countries do the Danube and the Rhine traverse?	`nn1`: countries `nnp1`: Danube `nnp2`: Rhine `prop1`: traverse
Template for logical difference questions		
`SELECT DISTINCT ?x1 WHERE {nnp1 prop1 ?x1 . ?x1 type_of ?x2 . ?x2 isa* nn1 . FILTER NOT EXISTS {nnp2 prop1 ?x1}}`	Which countries do the Danube but not the Rhine traverse?	`nn1`: countries `nnp1`: Danube `nnp2`: Rhine `prop1`: traverse

Templates for comparative questions Table 3 shows the SPARQL templates for Comparative questions. According to Table 1, there are six classes of Comparative questions. However, for the sake of space, here we show only two templates for them (i.e., for the Comparative Count over More and the Comparative More questions). In fact, the template for Comparative Count over More questions is very similar to the ones for Comparative Count over Less and Comparative Count over Approximately questions, since they only differ for the last comparison of values. For instance, in the case of Comparative Count over Less, it is sufficient to substitute the "$>$" symbol with "$<$". The same applies to the templates addressing Comparative Count Less and Comparative Approximately questions with respect to the Comparative Count More.

Furthermore, the two templates in Table 3 differ only for the overall counting in the first one.

Templates for quantitative questions In Table 4, some of the templates for Quantitative questions are shown. Analogously to Comparative questions, also, in this case, very similar templates are not listed. In fact, the templates for Quantitative Count over Atmost, Approximately, and Equal questions can be derived from the template for Quantitative Count over

Table 3 SPARQL templates for comparative questions

SPARQL template	Example of question	Query parameters
Template for comparative count over more questions		
SELECT (COUNT(?x21) AS ?x31) WHERE { SELECT ?x21 WHERE { {SELECT ?x21 (COUNT(?x23) AS ?x25) WHERE {?x21 type_of ?x22 . ?x22 isa* nn1 . ?x21 prop1 ?x23 . ?x23 type_of ?x24 . ?x24 isa* nn2} GROUP BY ?x21} {SELECT (COUNT(?x11) AS ?x13) WHERE {nnp1 prop1 ?x11 . ?x11 type_of ?x12 . ?x12 isa* nn2}} FILTER(?x25 >?x13)}}	How many states of the USA share borders with more states than California?	nn1: states of the USA prop1: share borders with nn2: states nnp1: California
Template for comparative more questions		
SELECT ?x21 WHERE { {SELECT ?x21 (COUNT(?x23) AS ?x25) WHERE {?x21 type_of ?x22 . ?x22 isa* nn1 . ?x21 prop1 ?x23 . ?x23 type_of ?x24 . ?x24 isa* nn2} GROUP BY ?x21} {SELECT (COUNT(?x11) AS ?x13) WHERE {nnp1 prop1 ?x11 . ?x11 type_of ?x12 . ?x12 isa* nn2}} FILTER(?x25 > ?x13)}	Which states of the USA share borders with more states than California?	nn1: states of the USA prop1: share borders with nn2: states nnp1: California

Atleast questions by modifying the condition expressed by means of the HAVING operator. The same considerations hold in the case of the templates for Quantitative Atmost, Approximately, and Equal questions with respect to the template for Quantitative Atleast questions. Finally, in order to obtain the template for Quantitative Min questions, it is sufficient to replace the MAX operator, with the MIN operator.

Template for simple questions As suggested by the name, this template is the simplest one and is shown in Table 5.

Table 4 SPARQL template for quantitative questions

SPARQL template	Example of question	Query parameters
Template for quantitative count questions		
`SELECT (COUNT(DISTINCT ?x1) AS ?x2) WHERE {nnp1 prop1 ?x1 . ?x1 type_of ?x3 . ?x3 isa* nn1}`	How many countries does the Danube traverse?	`nn1`: countries `nnp1`: Danube `prop1`: traverse
Template for quantitative count over atleast questions		
`SELECT ?x11 WHERE { {SELECT ?x11 (COUNT(?x13) AS ?x15) WHERE {?x11 type_of ?x12 . ?x12 isa* nn1 . ?x11 prop1 ?x13 . ?x13 type_of ?x14 . ?x14 isa* nn2} GROUP BY ?x11} FILTER(?x15 >= 3)}`	How many states of the USA share borders with at least 3 states?	`nn1`: states of the USA `prop1`: share borders with `nn2`: states
Template for quantitative atleast questions		
`SELECT ?x1 WHERE {?x1 type_of ?x2 . ?x2 isa* nn1 . ?x1 prop1 ?x3 . ?x3 type_of ?x4 . ?x4 isa* nn2} GROUP BY ?x1 HAVING(COUNT(?x3) >= 3)`	Which states of the USA share borders with at least 3 states?	`nn1`: states of the USA `prop1`: share borders with `nn2`: states

5 Experimental results

In this section, we first describe the dataset used in our experiments; then, we evaluate the performance of our system. We remind the reader that our system performs two main steps: (1) template-based question classification and (2) result retrieval. In the first step, given the NL question posed by the user, the system classifies it in order to choose the best query template. In the second step, the system uses the named entities, types, and relations extracted from the question to instantiate the selected template in order to obtain an actual SPARQL query. The query is run over the KB to get the result that will be returned to the final user. Hence, we decided first to evaluate the performance of the question classifier, then the performance of the end-to-end system that gets an NL question and answers it by using the KB as a source of information.

Table 4 continued

SPARQL template	Example of question	Query parameters
Template for quantitative max questions		
`SELECT DISTINCT ?x31 WHERE {?x31 type_of ?x32 . ?x32 isa* nn1 . ?x31 prop1 ?x33 . ?x33 type_of ?x34 . ?x34 isa* nn2 . {SELECT(MAX(?x15) AS ?x21) WHERE {{SELECT ?x11 (COUNT(?x13) AS ?x15) WHERE {?x11 type_of ?x12 . ?x12 isa* nn1 . ?x11 prop1 ?x13 . ?x13 type_of ?x14 . ?x14 isa* nn2} GROUP BY ?x11}}}} GROUP BY ?x31 ?x21 HAVING(COUNT(?x34) = ?x21)`	Which states of the USA share borders with the max number of states?	`nn1`: states of the USA `prop1`: share borders with `nn2`: states

Table 5 SPARQL template for simple questions

SPARQL template	Example of question	Query parameters
Template for simple questions		
`SELECT ?x WHERE {nnp1 prop1 ?x}`	Which is the capital of the USA?	**`prop1`**: capital **`nnp1`**: USA

Table 6 Statistics of selected CSQA dataset

	Num. of questions
Training set	516,012
Test set	130,905
Total	646,917

5.1 Dataset

For our experiments, we used the Complex-Sequential Question Answering (CSQA) dataset presented in [9]. It consists of NL questions divided into training, validation, and test sets of 1.5M, 167K, and 260K questions, respectively.

The questions are categorized into 45 different question classes (i.e., the description field in the dataset).

Table 7 Statistics of question classes from the selected data from the CSQA dataset

Class		Num. of questions
Logical	Union	57,021
	Intersection	13,907
	Difference	1748
Comparative	More	10,661
	Less	11,564
	Approx	12,088
Comparative count over	More	10,727
	Less	11,688
	Approx	12,004
Quantitative	Atmost	7104
	Atleast	6683
	Approx	12,903
	Equal	6019
	Max	16,490
	Min	16,203
Quantitative count		39,146
Quantitative count over	Atmost	7008
	Atleast	6571
	Approx	12,917
	Equal	6100
Simple Question		368,365
Total		646,917

From the CSQA dataset, we extracted questions that are categorized as Logical, Single Relation, Comparative/Quantitative/Simple Questions Single Entity type. We kept out the Indirect or Incomplete questions as they depend on the previous questions/answers of the conversation. These question types are outside the scope of our research since in this paper we do not address the problem of conversational Q/A. In any case, they can be easily included by using techniques such as co-reference resolution to propagate the subject previously mentioned in the conversation [44, 45]. After the filtering, the resulting dataset has a total of around 647K questions (see also Table 6). More detailed statistics on the number of questions for the selected classes are given in Table 7. As discussed in Sect. 3.2, some of the CSQA question classes may correspond to different query templates; therefore, we further split these categories into sub-classes. For example, the Comparative More/Less classes become two distinct query templates. After this refinement, we got a total of 21 classes each one corresponding to a different query template.

5.2 Evaluation

The evaluation is about two tasks: the question classification, where our main contribution lies, and the actual question answering, where templates are instantiated and queries are run over Wikidata.

5.2.1 Question classification

We experimented with three classification approaches using different sets of features. The classifiers are:

- *Naïve Bayes* It is a probabilistic machine-learning model used for classification tasks. It is based on the Bayes theorem, where given two variables A (the hypothesis) and B (the evidence), the probability of A happening, given that we know that B has occurred, is $P(A|B) = \frac{P(B|A) \times P(A)}{P(B)}$. The Naïve Bayes classifier makes the naïve assumption that the features (observed variables) are independent, therefore the presence of one particular feature does not affect the other [6].
- *Support Vector Machine (SVM)* It is a supervised machine-learning model that uses classification algorithms for two-group classification problems [7]. SVM takes data points and outputs the hyperplane that best separates the tags. This approach is largely employed in text classification tasks as it is fast and gives good performance even with a limited amount of data.
- *Gradient-Boosted Decision Trees (GDBT)* It is a nonparametric supervised learning method [5]. Gradient boosting identifies those machine-learning algorithms that combine many weak learners (in this case decision trees) together to create a stronger prediction model.

Regarding the features extracted from NL questions, we used syntactic and semantic features. For the former, we relied on PoS tagging performed by spaCy. Given a question, we extracted the PoS tags and used them to create a feature vector with the CountVectorizer function available in the Scikit-Learn package developed in Python.[7] Since syntactic features may fail to distinguish the content difference among questions with the same grammar structure, we also used semantic features that allow the differences in the semantic content of questions to be captured. By extracting unigrams (i.e., words) from questions, we get many unique words and, consequently, a sparse document-word matrix. Hence, we decided to perform a feature selection to identify only the more discriminating unigrams (e.g., *and, or, more, less, atleast, atmost,* etc.). Not surprisingly, very popular words (a.k.a., stopwords that are usually removed as not significant in IR and in topic modeling) are the most important ones to discriminate the different query templates.

Metrics The metrics used for our evaluation are accuracy, precision, recall, and F-score. The accuracy, A, is the number of questions for which the class is correctly predicted out of all the number of questions. The precision, P, is defined as the number of questions correctly classified as belonging to a given class out of the number of questions that truly belong to that class. Lastly, the recall, R, is the proportion of the questions that truly belong to one class and that are identified correctly. The F-score combines the precision and recall by computing the harmonic mean between them (i.e., $\frac{2 \times P \times R}{P+R}$). Since we are dealing with a multi-class problem, and the classes may be not perfectly balanced (i.e., some classes have more examples than others), we report both the macro-averaged metrics plus the weighted-average metrics.

Results The results using different classifiers are given in Table 8. On the top, we show the ones achieved with only syntactic features. As we can see, the best performance in terms of accuracy is achieved by GBDT and SVM. Inspecting the precision of different question classes, we notice that misclassifications were performed mostly in the sub-classes. This

[7] https://scikit-learn.org/stable/index.html.

Table 8 Experimental results using different classifiers and different sets of features

	Accuracy	Precision		Recall		F-score	
		Macro-Avg	Weigh-Avg	Macro-Avg	Weigh-Avg	Macro-Avg	Weigh-Avg
Only syntactic features							
Naïve Bayes	0.86	0.58	0.89	0.69	0.86	0.60	0.87
GBDT	0.91	0.68	0.91	0.67	0.91	0.65	0.90
SVM	0.93	0.73	0.93	0.71	0.93	0.69	0.92
Syntactic and semantic features							
Naïve Bayes	0.97	0.91	0.97	0.98	0.97	0.94	0.97
GBDT	0.99	0.99	1.00	0.99	1.00	0.99	1.00
SVM	1.00	1.00	1.00	1.00	1.00	1.00	1.00

Metrics are accuracy, precision, recall, and F-score (using macro- and weighted-average scores)

is expected as only syntactic features cannot discriminate similar questions that have the same grammar structure and differ only in the semantic content. As an example, given these two logical questions "Which countries are traversed by the Danube or the Rhine?" and "Which countries are traversed by the Danube and the Rhine?" we can see that after PoS tagging we get the same result "DET NOUN AUX VERB ADP DET PROPN CCONJ DET PROPON", however, these questions belong to two different query-template classes: Logical Union (CCONJ = or) and Logical Intersection (CCONJ = and), respectively. Therefore, we improved the classification performance by including semantic features. The results with syntactic and semantic features are shown at the bottom of Table 8. We observe that the GBDT and SVM classifiers get the best accuracy scores followed by Naïve Bayes.

In Table 9, we show the precision, recall, and F-score values for the single classes where predictions are obtained with the SVM classifier using only syntactic features (left) or both syntactic and semantic features (right). If we inspect the precision values in Table 9 (left), we can see that lower values of F-score are achieved with the Quantitative Approximate and Comparative Count over Approximate classes. By checking the confusion matrix, most of the incorrect predictions are among questions belonging to different classes but with a similar grammar structure. The precision improves consistently in Table 9 (right) where also semantic features are included, allowing to discriminate the adverbs (e.g., *equally* vs *approximately*) or adjectives (e.g., *more* vs *less*).

New patterns To further investigate the performance of the classifier, we checked how many new PoS patterns are in the test set compared to the training set. In the test set, out of 114K patterns, 16K are not present in the training set. Hence, 14% of patterns are completely new, nevertheless, the classifier can also handle these unseen PoS patterns and is able to assign the correct class by mapping them to the most similar observed patterns.

Table 9 Results achieved by the SVM classifier per single class

	Only syntactic features			Syntactic and semantic features		
	Precision	Recall	F-score	Precision	Recall	F-score
Log. Union	0.84	0.99	0.91	1.00	1.00	1.00
Log. Intersection	0.82	0.22	0.35	1.00	1.00	1.00
Log. Difference	0.97	0.99	0.98	1.00	0.99	1.00
Comp. Count. over More	0.63	0.29	0.39	1.00	1.00	1.00
Comp. Count. over Less	0.48	0.79	0.60	1.00	1.00	1.00
Comp. Count. over Approx	0.99	1.00	0.99	1.00	1.00	1.00
Comp. More	0.62	0.24	0.35	1.00	1.00	1.00
Comp. Less	0.49	0.81	0.61	1.00	0.97	0.98
Comp. Approx	0.99	0.98	0.99	1.00	1.00	1.00
Quant. Count	1.00	0.99	1.00	1.00	1.00	1.00
Quant. Count. over Atmost	0.81	0.71	0.76	1.00	1.00	1.00
Quant. Count. over Atleast	0.77	0.82	0.80	1.00	1.00	1.00
Quant. Count. over Approx	0.39	0.06	0.10	1.00	1.00	1.00
Quant. Count. over Equal	0.80	0.98	0.88	1.00	1.00	1.00
Quant. Atmost	0.73	0.73	0.73	1.00	1.00	1.00
Quant. Atleast	0.84	0.77	0.81	1.00	1.00	1.00
Quant. Approx	0.19	0.03	0.05	1.00	1.00	1.00
Quant. Equal	0.79	0.98	0.87	1.00	1.00	1.00
Quant. Max	0.63	0.85	0.73	1.00	1.00	1.00
Quant. Min	0.58	0.73	0.64	0.97	0.99	0.98
Simple Questions	1.00	0.99	1.00	1.00	1.00	1.00

Metrics are precision, recall, and F-score computed for each question class

5.2.2 End-to-end performance

We now describe our experiments for testing the performance of the end-to-end system. Due to the complexity and time-consuming experiment of querying the KB, for this evaluation, we decided to focus only on the Logical questions, and we created a subsample of them by taking the most frequent PoS patterns.

We used standard metrics adopted in IR: precision, recall, and F-score. The precision is the number of results that are returned by the system and that are truly relevant to the question over all the results reported by the system. The recall is the number of results provided by the system over the total number of results that are relevant to the question. F-score is the metric that summarizes the trade-off between precision and recall.

Results We extracted a sample from the CSQA dataset of Logical Union, Intersection, and Difference questions consisting of about 14K questions. For the subset of questions, we performed the retrieval of the answers and checked their correctness by using the CSQA replies as the ground truth. The results are shown in Table 10.

Table 10 End-to-end performance over a sample of the logical questions from the CSQA dataset averaged over the three classes

	Precision	Recall	F-Score
Logical union	0.80	0.83	0.80
Logical intersection	0.74	0.79	0.75
Logical difference	0.42	0.75	0.51
Average	0.65	0.79	0.69

Despite the precision of question classification being high, the retrieval precision can be low due to many difficulties that can be encountered when querying the KB, even when the SPARQL query template is correct. As a matter of fact, we noticed that, due to the follow-up updates of Wikidata, for some SPARQL queries, the entities, the entity types, or the relations, as well as the ids of the responses, differ from the ones in the CSQA dataset. This is particularly evident in the case of the Logical Difference questions where the precision significantly decreases with respect to the Logical Union and Logical Intersection questions. From these observations, we can conclude that the classification correctly identifies the query template whereas the instantiation fails due to some inconsistencies among the different versions of the KB. A possible way to overcome this limit in the evaluation is to perform the manual labeling of the answers. This is costly both in terms of time and money, and it requires the help of volunteers that must be qualified or trained for the labeling tasks. As an option, one can rely on crowdsourcing platforms such as Amazon Mechanical Turk or CrowdFlower. We could not perform such a labeling task as not all research institutions and universities allow research funds to be spent on crowdsourcing platforms due to bureaucratic issues.[8]

In Table 11, we show the values of precision, recall, and F-score metrics (when provided by the authors of the papers) for different approaches. We focus our attention on the template-based and machine-learning approaches that were tested on datasets with complex questions. We do not take into account approaches that handle simple questions, such as [16, 29], as the purpose of our work is to prove the ability of template-based approaches to handle complex questions. Also, we did not consider those works that focus only on a subset of replies such as specific classes of the KB [28].

We compare our approach with the one proposed by Saha et al. [9] as both were tested on the same data (i.e., the logical questions from the CSQA dataset) and the comparison can be fair. Although Saha et al. approach is completely automatic and does not require a pre-defined set of templates, it achieves a very low precision.

We also report the results of Bast and Hauffmann++ and QUINT [8], NEQA [26], and of a template-decomposition approach proposed by Zheng et al. [27]. All these approaches were tested on the ComplexQuestions dataset that was originally introduced by the authors of QUINT with the purpose of showing that their method can be trained with simpler single-clause questions (i.e., WebQuestions dataset [46]), and it can handle complex questions with multiple clauses. The ComplexQuestions benchmark is made of only 150 test questions and was manually created using the crawl of WikiAnswers. In particular, the authors asked a human annotator to collect questions with multiple clauses and to provide the gold standard answers. Bast and Haussmann++ is an enhanced version of the system *Aqqu*, originally proposed by Bast and Haussmann in [16]. The original version of the system deals only with simple questions; therefore in [8], the authors modified it with the purpose of handling com-

[8] In Italian public research institutes, it is very difficult to allocate money on a crowdsourcing experiment as it is not categorized as either software or hardware purchase.

Table 11 Performance of state-of-the-art approaches using datasets with complex questions

Method	Dataset	Precision	Recall	F-Score
Our approach	CSQA (logical) [9]	0.65	0.79	0.69
Saha et al. [9]		0.10	0.27	0.15
Bast and Haussmann++ [8, 16]		–	–	0.47
QUINT [8]	ComplexQuestions [8]	–	–	0.49
NEQA [26]		–	–	0.16
Zheng et al. [27]		–	–	0.71
Athreya et al. [15]	QALD-7 [47]	0.42	0.42	0.42
Dhandapani and Vadivel [25]	QALD-8 [48]	0.75	0.40	0.52
ARL [31]	WebQSP [49] (28% complex)	–	–	0.72
	CWQ [50] (64% complex)	–	–	0.43

plex questions. Each question was manually decomposed into its constituent sub-questions; then, the answer for each sub-question was computed using the original system *Aqqu*. Lastly, the QUINT stitching mechanism was applied to the answer sets of sub-questions to return an answer for the complete question. As we can see, the best-performing approach on the ComplexQuestions dataset is the one proposed by Zheng et al. [27]. Despite the good performance, it builds the templates automatically from the knowledge graph and the text corpus. Hence, the graph and the corpus have to be from the same source; otherwise, the generated templates may be limited because the entities in the graph may not be found in the corpus. Moreover, this approach was run on questions with multiple sentences connected by a grammar conjunction (i.e., "and", "or", "but") which is a only subset of the complex questions we considered from the CSQA dataset.

In Table 11, also [15] and [25] are present because, as already mentioned, the purpose of those works is similar to our proposal. The approaches were tested on QALD (question answering over linked data) benchmarks [47, 48]. The former relies on a recursive neural network method for question classification and does not address some of the SPARQL query operators; therefore, it cannot answer many of the questions shown in Table 1. The latter is based on strict assumptions, such as the completeness of the knowledge graph, which is not always feasible. In addition, we have included ARL [31] that, although based on a completely different approach, i.e., reinforcement learning, obtains a performance improvement in the case of the WebQuestionsSP (WebQSP) dataset [49], containing a small percentage of complex questions, whereas it remains on the state-of-the-art values when considering the ComplexWebQuestions (CWQ) dataset [50] which includes a large percentage of complex questions.

Note that Table 11 can help the reader get an insight into the current literature on complex Q/A over KBs. We would like to emphasize that, unfortunately, comparing the different approaches, even when belonging to the same architecture, is not feasible because of the use of different metrics, different datasets and, sometimes, the unavailability of the selected datasets and/or the resulting experimental values.

6 Conclusion

We have presented a semi-automatic approach for transforming NL questions into SPARQL queries to run over KBs (e.g., Wikidata). The approach first applies PoS tagging and keyword extraction to an NL question with the purpose of extracting features reflecting its grammatical structure and semantic content. Then, it classifies the question on the basis of these features into a class associated with a query template.

The set of proposed templates helps the automatic transformation from the NL questions to the SPARQL queries. Although our approach depends on a set of pre-defined query templates, it scales well for the classes of questions analyzed in this paper. In particular, the classifier can handle a large set of different PoS patterns, mapping them into classes corresponding to a few query templates. Moreover, it predicts the correct template also for unseen patterns, allowing the handling of new types of questions.

Note that template-based approaches either rely on a pre-defined set of query templates or automatically generate query templates from the NL questions. On the one hand, the approaches using a predefined set of query templates are limited to some types of questions, whereas our approach covers different types of NL questions with a relatively small set of templates. On the other hand, the automatic generation of query templates leads to an explosion in the number of templates and requires training data with question-answer pairs that are time-consuming and costly to gather, whereas our approach keeps small the set of query templates and does not need large training data with question-answer pairs.

With regard to the limitations of our approach, the overall performance of the end-to-end system depends on the quality of the named entity and relation linking. Indeed, in our experiments, we faced up the problem of using NER tools as it is not always possible to link the results of the named-entity recognition to the KB. Moreover, regarding the validation, for some of the queries, the entities, the relations, or the ids of the responses may differ from the ones of the KB due to the follow-up updates of Wikidata.

In future work, we plan to overcome these limitations and extend our approach to conversations as well as to further KBs.

Acknowledgements We gratefully acknowledge the partial support of the PNRR MUR project PE0000013-FAIR. We also thank the editor and the anonymous reviewers for their valuable comments.

Author Contributions All authors contributed to the research ideas, analyzed the data, and wrote the main manuscript text. The experiments were implemented by Ida Mele (data preprocessing, feature extraction, classification, and end-to-end experiments) and Francesco Taglino (SPARQL query template, querying the KB, and end-to-end experiments). Ida Mele prepared Fig. 1, Table 1, and Tables from 6 to 11. Francesco Taglino prepared Tables from 2 to 5. All authors reviewed the manuscript and contributed to the literature review.

Declarations

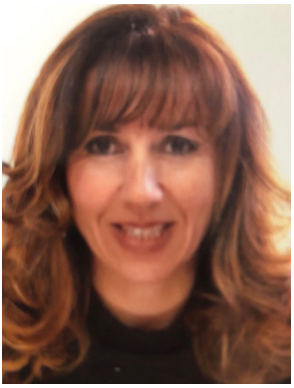
Conflict of interest The authors declare no competing interests.

References

1. Dimitrakis E, Sgontzos K, Tzitzikas Y (2020) A survey on question answering systems over linked data and documents. J Intell Inf Syst 55(2):233–259. https://doi.org/10.1007/s10844-019-00584-7
2. Ojokoh B, Adebisi E (2019) A review of question answering systems. J Web Eng 17(8):717–758. https://doi.org/10.13052/jwe1540-9589.1785
3. Voorhees EM (2001) Overview of the TREC 2001 question answering track. In: Proc. of the 10th text retrieval conference (TREC), pp 42–51
4. Pereira A, Trifan A, Lopes RP, Oliveira JL (2022) Systematic review of question answering over knowledge bases. IET Softw 16(1):1–13. https://doi.org/10.1049/sfw2.12028
5. Zheng Z, Zha H, Zhang T, Chapelle O, Chen K, Sun G (2008) A general boosting method and its application to learning ranking functions for web search. In: Proc. of advances in neural information processing systems, NIPS, pp 1697–1704
6. Friedman N, Geiger D, Goldszmidt M (1997) Bayesian network classifiers. Mach Learn 29(2):131–163
7. Cortes C, Vapnik V (1995) Support-vector networks. Mach Learn 20(3):273–297
8. Abujabal A, Yahya M, Riedewald M, Weikum G (2017) Automated template generation for question answering over knowledge graphs. In: Proc. of the 26th international conference on world wide web, WWW. ACM, New York, pp 1191–1200. https://doi.org/10.1145/3038912.3052583
9. Saha A, Pahuja V, Khapra MM, Sankaranarayanan K, Chandar S (2018) Complex sequential question answering: towards learning to converse over linked question answer pairs with a knowledge graph. In: Proc. of the 32nd AAAI conference on artificial intelligence and 38th innovative applications of artificial intelligence conference and 8th AAAI symposium on educational advances in artificial intelligence
10. Unger C, Freitas A, Cimiano P (2014) An introduction to question answering over linked data. In: Proc. of the 10th international summer school on reasoning on the web in the big data era. Springer, Cham, pp 100–140. https://doi.org/10.1007/978-3-319-10587-1_2
11. Mishra M, Mishra VK, Sharma H (2013) Question classification using semantic, syntactic and lexical features. Int J Web Semantic Technol 4(3):39
12. Metzler D, Croft WB (2005) Analysis of statistical question classification for fact-based questions. Inf Retr 8(3):481–504
13. Zhang D, Lee WS (2003) Question classification using support vector machines. In: Proc. of the 26th annual international ACM SIGIR conference on research and development in information retrieval. ACM, New York, pp 26–32
14. Dalton J, Fischer S, Owoicho P, Radlinski F, Rossetto F, Trippas JR, Zamani H (2022) Conversational information seeking: theory and application. In: Proc. of the 45th Int. ACM SIGIR conference on research and development in information retrieval. ACM, New York, pp 3455–3458. https://doi.org/10.1145/3477495.3532678
15. Athreya RG, Bansal SK, Ngomo A-CN, Usbeck R (2021) Template-based question answering using recursive neural networks. In: Proc. of IEEE 15th international conference on semantic computing (ICSC), pp 195–198. https://doi.org/10.1109/ICSC50631.2021.00041
16. Bast H, Haussmann E (2015) More accurate question answering on freebase. In: Proc. of the 24th ACM international on conference on information and knowledge management, CIKM '15. ACM, New York, pp 1431–1440. https://doi.org/10.1145/2806416.2806472
17. Figueroa A (2017) Automatically generating effective search queries directly from community question-answering questions for finding related questions. Expert Syst Appl 77:11–19. https://doi.org/10.1016/j.eswa.2017.01.041
18. Zhao S, Wang H, Li C, Liu T, Guan Y (2011) Automatically generating questions from queries for community-based question answering. In: Proc. of 5th international joint conference on natural language processing, pp 929–937
19. Fabbri A, Ng P, Wang Z, Nallapati R, Xiang B (2020) Template-based question generation from retrieved sentences for improved unsupervised question answering. In: Proc. of the 58th annual meeting of the association for computational linguistics, pp. 4508–4513. Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.acl-main.413
20. Abdelkawi A, Zafar H, Maleshkova M, Lehmann J (2019) Complex query augmentation for question answering over knowledge graphs. In: On the move to meaningful internet systems: OTM 2019 conferences—Confederated Int. Conferences: CoopIS, ODBASE, C&TC. LNCS, vol 11877, pp 571–587. https://doi.org/10.1007/978-3-030-33246-4_36
21. López V, Tommasi P, Kotoulas S, Wu J (2016) Queriodali: question answering over dynamic and linked knowledge graphs. In: Proc. of the 15th Int. semantic web conference, ISWC. LNCS, vol 9982, pp 363–382. https://doi.org/10.1007/978-3-319-46547-0_32

22. Singh K, Radhakrishna AS, Both A, Shekarpour S, Lytra I, Usbeck R, Vyas A, Khikmatullaev A, Punjani D, Lange C, Vidal M, Lehmann J, Auer S (2018) Why reinvent the wheel: let's build question answering systems together. In: Proc. of the 2018 world wide web conference on world wide web, WWW. ACM, New York, pp 1247–1256. https://doi.org/10.1145/3178876.3186023
23. Bakhshi M, Nematbakhsh M, Mohsenzadeh M, Rahmani AM (2020) Data-driven construction of SPARQL queries by approximate question graph alignment in question answering over knowledge graphs. Expert Syst Appl 146:113205. https://doi.org/10.1016/j.eswa.2020.113205
24. Hu S, Zou L, Yu JX, Wang H, Zhao D (2018) Answering natural language questions by subgraph matching over knowledge graphs. IEEE Trans Knowl Data Eng 30(5):824–837. https://doi.org/10.1109/TKDE.2017.2766634
25. Dhandapani A, Vadivel V (2022) Template-based question answering system over the semantic web. Int J Inf Retr Res 12(2):1–17. https://doi.org/10.4018/ijirr.300333
26. Abujabal A, Roy RS, Yahya M, Weikum G (2018) Never-ending learning for open-domain question answering over knowledge bases. In: Proc. of the 2018 world wide web conference, WWW. ACM, New York, pp 1053–1062
27. Zheng W, Yu JX, Zou L, Cheng H (2018) Question answering over knowledge graphs: question understanding via template decomposition. Proc VLDB Endow 11(11):1373–1386. https://doi.org/10.14778/3236187.3236192
28. Panchbhai A, Soru T, Marx E (2020) Exploring sequence-to-sequence models for SPARQL pattern composition. In: Proc. of the 2nd Iberoamerican conference and 1st Indo-American conference on knowledge graphs and semantic web KGSWC. Communications in computer and information science, vol 1232, pp 158–165. https://doi.org/10.1007/978-3-030-65384-2_12
29. Luo D, Su J, Yu S (2020) A BERT-based approach with relation-aware attention for knowledge base question answering. In: Proc. of Int. joint conference on neural networks, IJCNN, pp 1–8. https://doi.org/10.1109/IJCNN48605.2020.9207186
30. Yin X, Gromann D, Rudolph S (2021) Neural machine translating from natural language to SPARQL. Futur Gener Comput Syst 117:510–519. https://doi.org/10.1016/j.future.2020.12.013
31. Zhang Q, Weng X, Zhou G, Zhang Y, Huang JX (2022) ARL: An adaptive reinforcement learning framework for complex question answering over knowledge base. Inf Process Manag 59(3):102933. https://doi.org/10.1016/j.ipm.2022.102933
32. Gashkov A, Perevalov A, Eltsova M, Both A (2022) Improving question answering quality through language feature-based sparql query candidate validation. In: Proc. of the 19th Int. semantic web conference, ISWC. Springer, Berlin, pp 217–235. https://doi.org/10.1007/978-3-031-06981-9_13
33. Xiong G, Bao J, Zhao W, Wu Y, He X (2022) AutoQGS: Auto-prompt for low-resource knowledge-based question generation from SPARQL. In: Proc. of the 31st ACM international conference on information & knowledge management, CIKM. ACM, New York, pp 2250–2259. https://doi.org/10.1145/3511808.3557246
34. Yu Z, Yu J, Cui Y, Tao D, Tian Q (2019) Deep modular co-attention networks for visual question answering. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 6281–6290
35. Yu Z, Cui Y, Yu J, Tao D, Tian Q (2019) Multimodal unified attention networks for vision-and-language interactions. arXiv preprint arXiv:1908.04107
36. Voorhees EM, Tice DM et al (1999) The TREC-8 question answering track evaluation. In: Proc. of the 8th text retrieval conference (TREC), vol 1999, p 82
37. Li X, Roth D (2002) Learning question classifiers. In: Proc. of the 19th international conference on computational linguistics
38. Hovy E, Hermjakob U, Ravichandran D (2002) A question/answer typology with surface text patterns. In: Proc. of the human language technology conference (HLT), pp 247–251
39. Ittycheriah A, Franz M, Zhu W-J, Ratnaparkhi A, Mammone RJ (2000) IBM's statistical question answering system. In: Proc. of the 9th text retrieval conference (TREC)
40. Chinchor N, Robinson P (1997) MUC-7 named entity task definition. In: Proc. of the 7th conference on message understanding, vol 29, pp 1–21
41. Nyberg E, Mitamura T, Callan JP, Carbonell JG, Frederking RE, Collins-Thompson K, Hiyakumoto L, Huang Y, Huttenhower C, Judy S et al (2003) The JAVELIN question-answering system at TREC 2003: a multi-strategh approach with dynamic planning. In: Proc. of the 12th text retrieval conference (TREC), pp 2–1
42. Harris ZS (1954) Distributional structure. Word 10(2–3):146–162
43. Le Q, Mikolov T (2014) Distributed representations of sentences and documents. In: Proc. of the international conference on machine learning, pp 1188–1196

44. Mele I, Muntean CI, Nardini FM, Perego R, Tonellotto N, Frieder O (2020) Topic propagation in conversational search. In: Proc. of the 43rd international ACM SIGIR conference on research and development in information retrieval. ACM, New York, pp 2057–2060
45. Mele I, Muntean CI, Nardini FM, Perego R, Tonellotto N, Frieder O (2021) Adaptive utterance rewriting for conversational search. Inf Process Manag 58(6):102682
46. Berant J, Chou A, Frostig R, Liang P (2013) Semantic parsing on freebase from question-answer pairs. In: Proc. of the 2013 conference on empirical methods in natural language processing. EMNLP'13, pp 1533–1544
47. Usbeck R, Ngomo A-CN, Haarmann B, Krithara A, Röder M, Napolitano G (2017) 7th Open challenge on question answering over linked data (QALD-7). In: Semantic web challenges. Springer, Cham, pp 59–69
48. Usbeck R, Ngomo AN, Conrads F, Röder M, Napolitano G (2018) 8th Challenge on question answering over linked data (QALD-8) (invited paper). In: Joint Proc. of SemDeep-4, NLIWOD-4, QALD-9 Co-located with 17th international semantic web conference (ISWC 2018). CEUR workshop proceedings, vol 2241, pp 51–57
49. Yih W, Richardson M, Meek C, Chang M-W, Suh J (2016) The value of semantic parse labeling for knowledge base question answering. In: Proceedings of the 54th annual meeting of the association for computational linguistics (volume 2: short papers). Association for Computational Linguistics, Berlin, pp 201–206. https://doi.org/10.18653/v1/P16-2033
50. Talmor A, Berant J (2018) The web as a knowledge-base for answering complex questions. In: Proceedings of the 2018 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long papers). Association for Computational Linguistics, New Orleans, pp 641–651. https://doi.org/10.18653/v1/N18-1059



Anna Formica received the degree (Hons.) in mathematics from Sapienza University of Rome in 1989. She is currently a Senior Researcher with the "Istituto di Analisi dei Sistemi ed Informatica" (IASI) "Antonio Ruberti" of the Italian National Research Council (Consiglio Nazionale delle Ricerche-CNR), Rome, where she leads the "Software and Knowledge-Based Systems" (SaKS) Group. She took part in various research projects of the European framework programs and bilateral projects with international institutions. Her current research interests include semantic web, similarity reasoning, formal specification and validation of domain ontologies, fuzzy formal concept analysis, geographical information systems, and e-learning. She serves as a referee for several international journals and conferences.

Ida Mele is currently a researcher at the "Istituto di Analisi dei Sistemi ed Informatica" (IASI) "Antonio Ruberti" of the Italian National Research Council (Consiglio Nazionale delle Ricerche - CNR) in Rome, as member of the "Software and Knowledge-based Systems" (SaKS) group. Previously, she was a postdoctoral researcher at ISTI-CNR in Pisa (Italy), the University of Lugano (Switzerland), and MPII in Saarbruecken (Germany). She received her Bachelor's and Master's Degrees (Hons.) in Computer Engineering from Sapienza University of Rome. In 2014, she got her Ph.D. in Computer Engineering from Sapienza University of Rome with a Thesis on Web Usage Mining with applications to Web search and recommendation. Part of her Ph.D. research was carried out during internships at Yahoo Research (Spain) and MPII (Germany). She has co-authored papers in peer-reviewed conferences and in top-tier journals. She also serves as a PC member and reviewer for several international conferences and journals. Her research interests include Web Mining, Information Retrieval, Q/A Systems, and Conversational Assistants.

Francesco Taglino was graduated in Information Science at the Sapienza University of Rome in 1999. Since 2009, he is a permanent researcher at the "Istituto di Analisi dei Sistemi ed Informatica" (IASI) "Antonio Ruberti" of the Italian National Research Council (Consiglio Nazionale delle Ricerche - CNR) in Rome, as member of the "Software and Knowledge-based Systems" (SaKS) group. His main research interests are on knowledge representation and reasoning and semantic technologies, and in particular on ontology engineering, semantic similarity and relatedness, as well as quantum computing. He participated in several national and international projects, mainly in the context of enterprise interoperability, and serves as a referee in several international journals and conferences.