SmartBench: Demonstrating Automatic Generation of Comprehensive Benchmarks for Question Answering Over Knowledge Graphs

Abdelghny Orogat
Carleton University
abdelghny.orogat@carleton.ca

Ahmed El-Roby
Carleton University
ahmed.elroby@carleton.ca

ABSTRACT

In recent years, a significant number of question answering (QA) systems that retrieve answers to natural language questions from knowledge graphs (KG) have been introduced. However, finding a benchmark that accurately evaluates the quality of a question answering system is a difficult task because of (1) the high degree of variations with respect to the fine-grained properties among the available benchmarks, (2) the static nature of the available benchmarks versus the evolving nature of KGs, and (3) the limited number of KGs targeted by existing benchmarks, which hinders the usability of QA systems in real deployment over KGs that are different from those which the QA system was evaluated using. In this demonstration, we introduce SmartBench, an automatic benchmark generating system for QA over any KG. The benchmark generated by SmartBench is guaranteed to cover all the properties of the natural language questions and queries that were encountered in the literature as long as the targeted KG includes these properties.

PVLDB Reference Format:
Abdelghny Orogat and Ahmed El-Roby. SmartBench: Demonstrating Automatic Generation of Comprehensive Benchmarks for Question Answering Over Knowledge Graphs. PVLDB, 15(12): 3662 - 3665, 2022.
doi:10.14778/3554821.3554869

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/aorogat/SmartBench.

1 INTRODUCTION

The number of knowledge graphs (KGs) has increased at an unprecedented rate in recent years [2, 6, 9]. These knowledge graphs include a wealth of information that can be potentially utilized for question answering (QA). Finding answers in a KG, on the other hand, is a difficult task. To articulate questions in a structured format that can be utilized to identify matches in the KG, the user must have a thorough understanding of both the KG and a structured query language. This restriction limits the ability to ask questions to power users who can compose syntactically and semantically correct queries that appropriately describe their information needs. Such power users make up a small percentage of a potentially large user base. To address this issue, a wide variety of QA systems have been developed to help non-expert users to specify their information needs in natural language [3].

Several benchmarks were introduced to evaluate QA systems [7, 8]. These benchmarks typically include natural language questions, answers to these questions from the KG targeted by the benchmark, and possibly structured queries that can be used to retrieve these answers. To evaluate a new QA system, its developers must select a subset from a number of available benchmarks (at least 17 [5]). Without a quantitative comparison that emphasizes the differences between these benchmarks, selecting a subset of them to evaluate a new QA system is motivated primarily by the ease of comparison to existing systems in the literature rather than by the effectiveness of a benchmark in evaluating a QA system. In fact, existing benchmarks differ significantly from each other. In light of [4, 5], we would like to highlight the following three main issues:

- **Variations among benchmarks:** Although all benchmarks target the same problem and sometimes the same KG, there are high-degree variations with respect to several linguistic features in the natural language questions , and syntactical and structural features in the queries. For example, QALD-9 [8] does not have *Whom*, *Whose*, and *Topical* natural language questions while LC-QuAD-1 [7] does. Another example is that QALD-9 has better query shape coverage than LC-QuAD-1 and has more complex questions that correspond to more complex query shapes like Cycle and Flower shapes (will be discussed in Section 2). These variations indeed affect the reported quality of QA systems to the degree that one QA system can be shown to be better or worse than other QA systems based only on the change of the used benchmark [5].
- **Targeting a limited number of KGs:** The existing benchmarks target only a handful of KGs with more focus on larger cross-domain KGs (e.g., DBpedia [1]). Such settings assume that using such KGs to evaluate QA systems may be an indication of how well they will perform over smaller or domain-specific KGs. However, our experiments show that most QA systems either do not support portability, do not exhibit similar performance to the one reported using existing benchmarks, or both. Indeed, in practical deployments, the targeted KG is a proprietary enterprise dataset (e.g., deploying a chat bot to answer customer/user questions). In such settings, the performance of existing QA systems that were evaluated using the existing benchmarks over a different target KG is questionable.
- **Staleness:** KGs are continuously evolving, while benchmarks are static by nature. Most of the KGs now have more recent

Proceedings of the VLDB Endowment, Vol. 15, No. 12 ISSN 2150-8097.
doi:10.14778/3554821.3554869

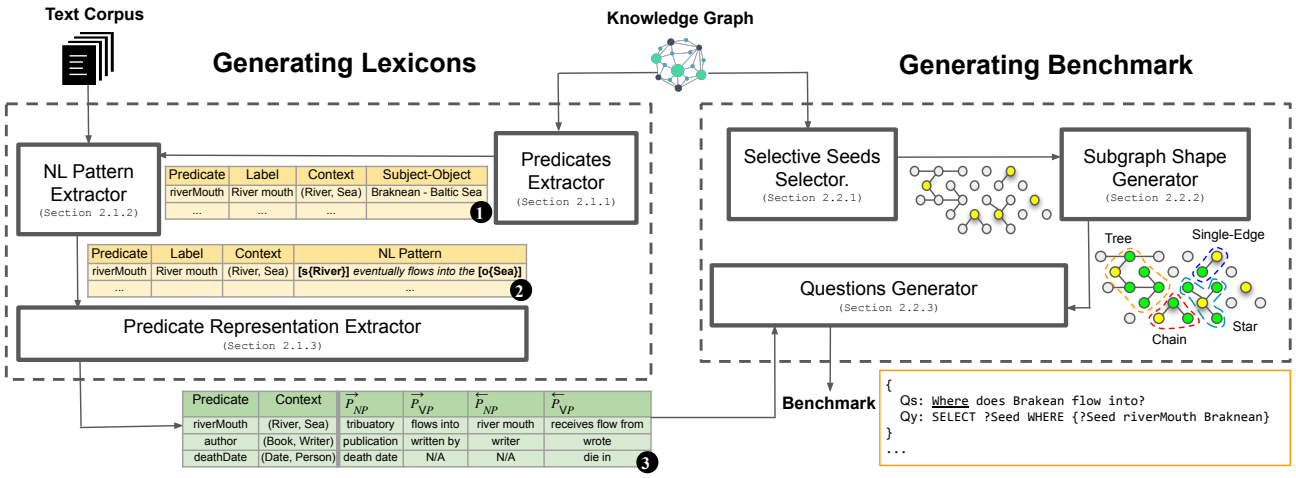

Figure 1: SmartBench Architecture.

versions than those that were used to generate/create the benchmark. Updating the benchmark answers based on the live KG will not work if the KG's evolution touches its ontology or vocabulary. In fact, we notice such behavior in existing benchmarks. Some queries no longer retrieve answers because the ontology of DBpedia has changed. Some other queries in earlier QALD benchmarks are no longer syntactically correct due to changes in the standard query language (SPARQL). The invalidation of such entries in the benchmarks also invalidates the aspects of assessments that these questions were intended to target in the evaluated QA systems.

In this demonstration, we showcase SmartBench[1], a benchmark generating system that addresses the aforementioned issues. SmartBench guarantees that the generated benchmark comprehensively covers syntactic and structural properties of the targeted KG in its questions[2]. SmartBench can be used to generate a benchmark over *any* KG as long as the KG includes meaningful labels of its entities and predicates. Moreover, using an external text corpus, SmartBench can capture semantically-equivalent utterances that can be used to ask questions in different ways to better challenge the QA system.

2 OVERVIEW OF SMARTBENCH

Figure 1 shows the architecture of SmartBench, which has two main phases: (1) Generating Lexicons, where SmartBench generates semantically-rich predicates lexicons and (2) Generating Benchmark, where SmartBench utilizes the generated lexicons and the KG to generate the benchmark questions, queries, and answers.

2.1 Generating Lexicons

2.1.1 Predicates Extractor. The Predicates Extractor scans the KG to extract all its predicates, their labels, their contexts (i.e., the type of the subject and the type of the object connected by the predicate), and randomly sample a number of triples for each context. To avoid noisy contexts (i.e., contexts that have a relatively small number of entities associated with them in comparison to other contexts for the same predicate), we exclude the least frequent contexts in the KG. Table ❶ shows an example of the output of this step.

2.1.2 Natural Language Pattern Extractor. The output of the Predicates Extractor can be used to generate the predicates lexicons using the labels of the predicates and entities. However, the output lexicons will not capture the various ways to describe semantically-equivalent utterances using natural language. Therefore, SmartBench can optionally utilize an external text corpus to extract all the sentences in which the (subject, object) pair of each predicate-context pair appear together. For example, using the triple example *(Braknean, riverMouth, Baltic_Sea)* of the predicate-context pair (*riverMouth*, (*River* , *Sea*)) from DBpedia, we could extract the sentence "`Braknean eventually flows into the Baltic Sea`". Consequently, SmartBench replaces the subject and the object in the sentence with *[s{S_type}]* and *[o{O_type}]*, respectively, to produce the natural language pattern "`[s{River}] eventually flows into the [o{Sea}]`". This natural language pattern can be utilized for all the instances of the *(River, Sea)* context. The output of this step is shown in Table ❷.

2.1.3 Predicate Representation Extractor. In practice, the natural language patterns are not as simple as the one discussed in Section 2.1.2. The extracted sentences can be long and can contain multiple verb and noun phrases, resulting in a difficulty in finding crisp noun or verb phrases for the given predicate-context pairs. For example, the pattern "`[s{Book}] (also known as girls no more)[1] is a 1986 fiction novel written by [o{Writer}]`" is extracted by SmartBench for the predicate-context pair (*author*, (*Book* , *Writer*)). This sentence pattern has two verb phrases such as "`known as`" and "`written by`" and one noun phrase like "`a 1986 fiction novel`". In SmartBench, we differentiate between four phrase representations based on the phrase type (Verb Phrase or Noun Phrase) and the relationship direction (subject-to-object or object-to-subject). We use $\overrightarrow{P}_{NP}$, $\overrightarrow{P}_{VP}$, $\overleftarrow{P}_{NP}$, and $\overleftarrow{P}_{VP}$ to refer to these phrase representations. We break down long sentences into verb/noun phrases whose direction is determined by whether the

[1]SmartBench is publicly available at https://github.com/aorogat/SmartBench
[2]More details on these properties can be found in [5]

subject/object is mentioned first in the sentence. Eventually, we end up with several phrase candidates for each phrase representation ($\overrightarrow{P}_{NP}$, $\overrightarrow{P}_{VP}$, $\overleftarrow{P}_{NP}$, and $\overleftarrow{P}_{VP}$) for each predicate-context pair. We choose the top candidate based on an embeddings-based ensemble similarity function that takes into account (1) the similarity between the phrase and the literal of the predicate (e.g., *written by* and *author*), (2), the similarity between the phrase and the type of the subject (e.g., *written by* and *book*), (3) the similarity between the phrase and the type of the object (e.g., *written by* and *writer*), and (4) the normalized frequency of the phrase tokens in their base form (e.g., *write* instead of *written by*) in all the extracted sentences for the predicate-context pair. Table ❸ shows an example output.

2.2 Generating Benchmark

2.2.1 Selective Seeds Selector. The objective of this step is to select the entities that will serve as the answers (or subsets of the answers) to the generated questions, which we refer to as seed entities, or seeds for short. SmartBench does not randomly choose the seeds. We rather aim at selecting seeds that have good coverage of the classes in the KG. However, since we are limited by a user-input number of questions to generate (user input is discussed in Section 2.3), SmartBench selectively picks the entities whose classes are more common (large number of entities belong to the class). SmartBench first starts by implementing a widening step approach to select the classes to target such that more common classes are selected more frequently, while not forgoing least common classes. The size of the widening step is a function of the user-input required number of questions in the benchmark to ensure having a good representation of classes in the benchmark. For each of the selected classes, a similar approach is applied to select the seeds such that the seeds that are connected to more nodes (i.e., more significant in the KG) are selected more frequently. The output of the selective seeds selector is the set of entities **E** starting from which several subgraphs shapes will be generated. These subgraphs will form the shape of the query corresponding to a question in the benchmark.

2.2.2 Subgraph Shape Generator. The subgraph shape generator traverses the KG starting from the seed entities **E** to extract a set of subgraphs *SG* of different shapes. SmartBench is guaranteed to extract subgraphs that span all the different shapes encountered in the literature [5] as long as they exist in the KG. Some of these shapes are shown in the top part of Figure 2. Following, we summarize the shapes generated by SmartBench:

Single-Edge. This is the simplest shape that can be generated, which is a single triple pattern in the form of $(Seed, P, O)$ or $(S, P, Seed)$. The top part of Figure 2a shows an example of this case.

Chain. The chain shape consists of a series of connected single-edge shapes in the form (S_1, P_1, O_1), (O_1, P_2, O_2) ,..., (O_{n-1}, P_n, O_n), where *Seed* can either replace S_1 or O_n. The top part of Figure 2b shows an example of this case.

Star. The seed is the central node in this shape such that it is the common node among a set of chains. The top part of Figure 2c shows an example of this case.

Tree. The tree shape is more complex than the star shape, where it can be defined recursively as a star of stars. The seed is considered to be the root of the tree.

Cycle. The cycle is a special case of the chain shape, where the seed is the first and last node in the chain. That is, $(Seed, P_1, O_1)$, (O_1, P_2, O_2) ,..., $(O_{n-1}, P_n, Seed)$. The top part of Figure 2d shows an example of this case.

Flower. The flower shape is a subgraph with a node that is connected to at least one attachment that could have any of the following shapes: Single-Edge, Chain, Cycle, and Star.

Set-shapes. The set-shapes consist of two or more unconnected shapes. This shape is mostly used for comparisons between the unconnected shapes. For example, the question `"Which Company has employees more than Apple Inc.?"` compares between the number of employees in all companies and number of employees in Apple Inc., where in the general case, there are no connections between Apple Inc. and these companies.

2.2.3 Questions Generator. The questions generator utilizes the output of the predicate representation extractor (Table ❸ in Figure 1) and linguistic heuristics to generate natural language questions for the single-edge shape subgraphs. The focus on such shape is derived by the fact that the questions representing all the other shapes can be incrementally composed of two or more single-edge utterances using coordinating conjunctions.

We rely on linguistic heuristics to determine the question type, which depends on the type of the seed. When the seed is of type *Person*, we use the *Who*, *Whom*, or *Whose* keywords to ask about the seed based on selecting ($\overrightarrow{P}_{VP}$ or $\overrightarrow{P}_{NP}$), $\overleftarrow{P}_{VP}$, or $\overleftarrow{P}_{NP}$, respectively. When the seed is of type *Place*, we use the *Where* or *What* keywords to ask about the seed based on selecting $\overleftarrow{P}_{VP}$ or $\overrightarrow{P}_{NP}$, respectively. For all other types of entities, the *What* keyword is used. Finally, *How-Adj* and *When* can be used with numbers and dates, respectively. Figure 2a shows an example of a *What* question using $\overleftarrow{P}_{NP}$ for the predicate-context pair $(riverMouth, (River\ , Sea))$.

For generating questions for more complex shapes, SmartBench relies on using coordinating conjunctions and replacing an entity in the subgraph shape with its single-edge utterance from the output of the predicate representation extractor. For example, in Figure 2b, the Baltic sea in the question *What is the outflow of the Baltic sea?* is replaced by the noun phrase that is used to generate a question whose seed is the Baltic sea. Figure 2c shows an example of using the *and* coordinating conjunction.

2.3 User Interface

Figure 3 shows the user interface of SmartBench where the user can input ❶ the name and URL of the KG, ❷ the text Corpus, and ❸ the benchmark parameters.

For ❶, we support all Knowledge Graphs that accept HTTP GET or POST requests to retrieve the data via SPARQL queries. SmartBench uses the labels of the entities and the predicates in the KG to construct the questions as illustrated in Section 2. SmartBench can also optionally utilize a text corpus to capture natural language variations to represent semantically equivalent utterances. In ❷, the users can specify the Text Corpus source. Currently, we support Wikipedia and Google Search API (to retrieve documents with mentions of entities of interest). Finally, the users inputs the benchmark parameters in ❸. The Maximum Answer Cardinality is used to exclude questions that have a number of answers that is larger

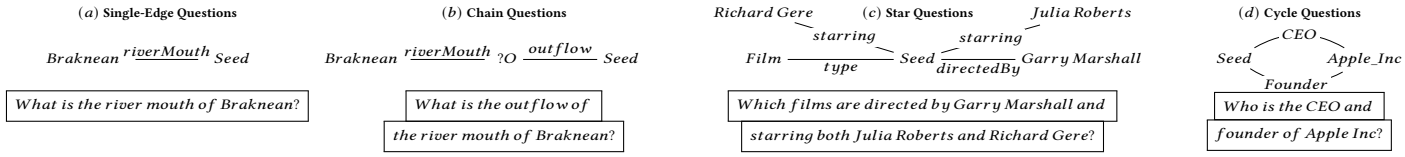

Figure 2: Question generation based on the subgraph Shape.

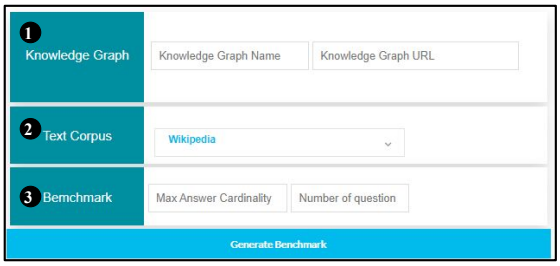

Figure 3: The user interface of SmartBench.

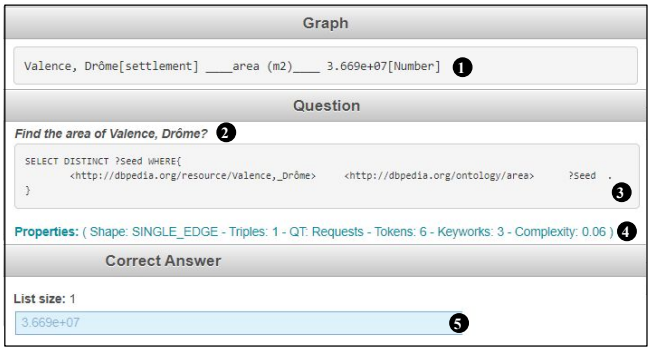

Figure 4: A sample Question generated by SmartBench over DBpedia.

than the user's input. The number of questions specifies how many questions the user would like to output in the benchmark.

2.4 Use Case of DBpedia

Figure 4 shows a sample output of generating a benchmark that targets DBpedia. First, SmartBench retrieves a seed entity (i.e., Valence, Drôme) which is used to generate the subgraph ❶ (single-edge). The subgraph is used to generate the natural language question ❷, whose corresponding query is ❸. The properties of the subgraph, natural language question, and the query are shown in ❹. ❺ is the correct answer retrieved by the query ❸.

Figure 5 shows a comparison that can be viewed in SmartBench between the benchmark generated by SmartBench, QALD-9, and LC-QuAD-1 with respect to the percentage of question types in all the benchmarks. The figure shows that the benchmark generated by SmartBench covers all the question types, which is not the case for the other benchmarks. Similar figures that show other comparisons with more benchmarks with respect to other syntactical (e.g., query keywords and operators) and structural features (query shapes) can also be viewed in SmartBench.

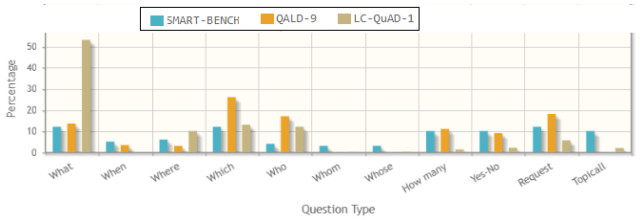

Figure 5: The frequency of question types generated over DBpedia for SmartBench, QALD-9, and LC-QuAD-1.

3 DEMONSTRATION SCENARIO

In this demonstration, the participants will select from a set of KGs for which SmartBench will generate benchmarks. This set of KGs span KGs that were previously used in the literature as well as KGs for which a benchmark was never generated. The participants can then browse the generated questions and do the following:

- Assess the quality of the automatically-generated natural language questions.
- Examine the complexity of the questions and associate this complexity with the shape of the corresponding queries.
- For the benchmarks generated for the KGs that were previously targeted in the literature, the participants can compare the differences in the fine-grained properties of the natural language questions and queries. It will be apparent that the benchmarks generated by SmartBench cover more properties.
- SmartBench will be prepackaged with several QA systems such that these systems are evaluated using the benchmark generated by SmartBench and other benchmarks from the literature. The participants will be able to compare the reported quality scores of the QA systems and observe their high-degree variations.

REFERENCES

[1] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. Ives. DBpedia: A nucleus for a web of open data. In *ISWC*. 2007.
[2] A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E. R. Hruschka, and T. M. Mitchell. Toward an architecture for never-ending language learning. In *AAAI*, 2010.
[3] K. Höffner, S. Walter, E. Marx, R. Usbeck, J. Lehmann, and A.-C. Ngonga Ngomo. Survey on challenges of question answering in the semantic web. *Semantic Web*, 8(6), 2017.
[4] A. Orogat and A. El-Roby. CBench: Demonstrating Comprehensive Evaluation of Question Answering Systems over Knowledge Graphs Through Deep Analysis of Benchmarks. *PVLDB*, 14(12), 2021.
[5] A. Orogat, I. Liu, and A. El-Roby. CBench: Towards Better Evaluation of Question Answering Over Knowledge Graphs. *PVLDB*, 14(8), 2021.
[6] F. M. Suchanek, G. Kasneci, and G. Weikum. Yago: a core of semantic knowledge. In *WWW*, 2007.
[7] P. Trivedi, G. Maheshwari, M. Dubey, and J. Lehmann. LC-QuAD: A corpus for complex question answering over knowledge graphs. In *ISWC*, 2017.
[8] R. Usbeck, R. H. Gusmita, M. Saleem, and A.-C. N. Ngomo. 9th challenge on question answering over linked data (QALD-9). *Joint Workshop on NLIWoD and Question Answering over Linked Data challenge*, 2018.
[9] D. Vrandečić and M. Krötzsch. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10), 2014.