How to Revert Question Answering on Knowledge Graphs

Gaurav Maheshwari[1*], Mohnish Dubey[1*], Priyansh Trivedi[1*], and Jens Lehmann[1,2]

[1] University of Bonn, Germany
{gaurav.maheshwari, priyansh.trivedi}@uni-bonn.de {dubey, jens.lehmann}@cs.uni-bonn.de
[2] Fraunhofer IAIS, Bonn, Germany
jens.lehmann@iais.fraunhofer.de

Abstract. A large scale question answering dataset has a potential to enable development of robust and more accurate question answering systems. In this direction, we introduce a framework for creating such datasets which decreases the manual intervention and domain expertise traditionally needed. We describe in details the architecture and the design decision we took while creating the framework.

1 Introduction

Knowledge graphs, such as DBpedia [5], Freebase [1], and Wikidata [10] contain large amounts of interesting information. However, this information can often only be queried by users familiar with query languages and the structure of the knowledge graph. To migitate this problem, numerous question answering (QA) approaches for knowledge graphs have been devised (see [4]). Many of them rely on machine learning techniques, which require large amounts of labeled training data. In this direction, we introduced LC-QuAD[7] (Large-Scale Complex Question Answering Dataset). LC-QuAD consists of 5000 natural language questions (NLQ) along with the intended SPARQL queries required to answer questions over DBpedia[5]. In this article, we describe the framework used for creating LC-QuAD and how it can be applied to other data sources.

2 Dataset Generation Workflow

Traditionally, question answering datasets have been created by manually converting a set of questions to their logical forms (for instance, [9], [2]). Although a dataset created in this manner can have varied questions, it requires substantial manual work from domain experts, and hence is not scalable. Thus, the need for an alternate workflow with (i) lesser human intervention and (ii) reduced domain expertise is felt.

In this direction, we present a framework for generating questions and their corresponding logical forms. We created our framework by reverse engineering the architecture of Semantic Parsing based Question Answering systems (SQA) such as [3] and [8]. These systems convert an NLQ to a formal query language expression, whereas we start

*These three authors contributed equally.

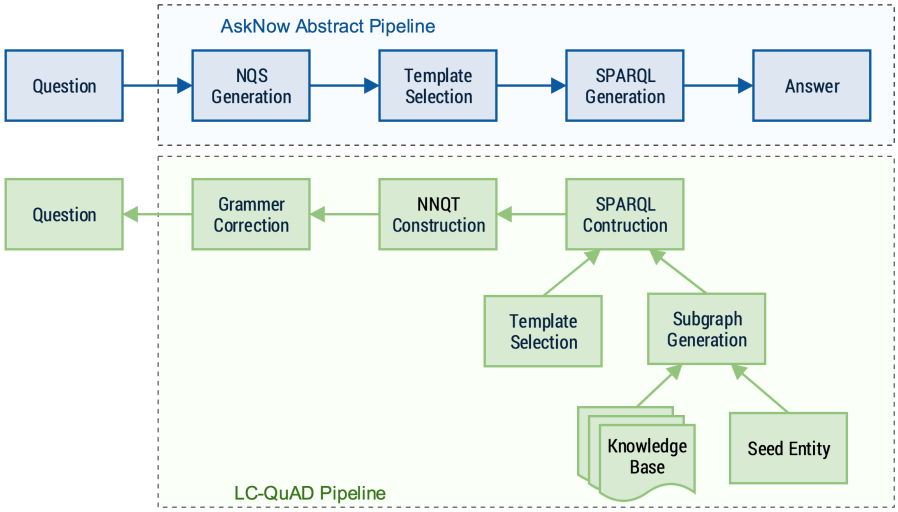

Fig. 1. The architecture of our framework in contrast to that of AskNow[3]

with a formal query language expression and convert it to an NLQ. The reverse task is easier because formal query languages have well defined semantics, and the entities and predicates occurring in the query are explicitly mentioned. Moreover, the target language (NL) is much more resilient to minute errors.

We now describe how we reverse engineered the architecture of AskNow[3] to create a question generation framework. Figure 1 gives an overview of architectures of both of them. Throughout the description, we will use the question: *"Name the capital of Germany?"* as our running example, to elaborate our dataset generation process in correspondence with the process of answering this question.

AskNow[3] breaks down the process of answering questions into two parts: conversion of questions into an expression of a semi-formal language, namely Normalized Query Structure (NQS) and subsequently converting the NQS expression to that of a formal language. NQS acts like a surface level syntactic template for queries, which maps NLQs having different syntactic structures into a common structure. Given the question, AskNow uses rules based on NL features to create a corresponding NQS instance. Thereafter it maps the entity (*dbr:Germany*), and the predicate (*dbo:capital*) present in the question to resources in the target KB, i.e. DBpedia. After mapping, the NQS instance is converted into SPARQL using machine generated and hand made rules. Finally, the query is executed on the query endpoint of *DBpedia* to retrieve the required answer: *dbr:Berlin*.

We begin our process where AskNow ends, i.e. collecting the answer of the questions we aim to generate. For our framework, a list of these answers act as seed entities[3]. Let "*dbr:Berlin*" be one such entity. We then create SPARQL queries which would all have this entity as one of their answers. To do so, we generate a subgraph of all the triples within 2-hops around these entities. The triple <*dbr:Germany dbo:capital dbr:Berlin*> would be one such triple in the subgraph. Since the number of such triples increases

[3]https://figshare.com/articles/Seed_Entities/5008286

exponentially with increasing hops, we reduce the size of subgraph, by stochastically removing predicates. We also remove every triple whose predicates do not occur in our whitelist[4]. We then juxtapose triples of this subgraph onto a list of SPARQL templates[5] to generate a set of valid SPARQL queries, all of them having *dbr:Berlin* as one of its answers. For instance, `SELECT ?uri WHERE { dbr:Germany dbo:capital ?uri. }` Note that while in our example, the SPARQL query comprises of only one triple, we can generate queries having two or more triples in the same manner.

At this point, instead of converting the SPARQL queries to NLQ directly, we employ a two step process similar to AskNow. This helps in further reducing manual intervention required for this conversion. We first transform the queries to an Normalized Natural language Question Templates (NNQT) instance. NNQTs are NLQ templates, corresponding to a SPARQL template with placeholders to be replaced with the surface form of entities and predicates present in the query. They interpret the semantics of its corresponding SPARQL template and verbalizes it, thereby transforming our aforementioned problem to that of converting semi-correct (grammatically) NLQ to a grammatically correct question. These NNQT's are KB independent and thus can be appropriated for any target KB. An NNQT instance for our running example would look like: $What\ is\ the\ \langle\ capital\ \rangle\ of\ \langle\ Germany\ \rangle?$

As is evident, this NNQT instance can be easily converted to a grammatically correct NLQ by native English speakers, without understanding the SPARQL syntax or any knowledge of the underlying KB schema. This allows our process to scale with minimal efforts. The person correcting these questions is also expected to paraphrase the questions, in order to increase the diversity of our dataset. The resultant NLQ of our NNQT instance would at this point be- "*Name the capital of germany ?*" Finally, an independent reviewer reviews every corrected question, and is expected to make minute tweaks and edits in case of errors. The reviewer is not expected to paraphrase the queries, thereby significantly reducing the time required in this step. The final output of our process would be - "*Name the capital of Germany?*"

Throughout the process, we use numerous techniques to increase the diversity and complexity of the questions so generated. Some of these techniques are: (i) replacing the entity surface forms within the NNQT instances with their synonyms, using WordNet[6], (ii) similarly, using Wikidata[10] for predicate surface forms, (iii) encouraging the question corrector to paraphrase the questions, keeping their semantics intact, (iv) declaring multiple NNQTs for every SPARQL template and stochastically selecting one of them. Due to the modularity of our framework, more such techniques can be added in the future.

3 Conclusions

In this article, we described a framework for generating QA datasets having questions and their equivalent logical forms. The general design of this framework follows a reverse process of SQAs, and in effect reduces the efforts involved in creating questions. We have successfully used our framework to create a dataset, LC-QuAD[7], having 5000

[4]https://figshare.com/articles/White_List_Relations/5008283

[5]https://figshare.com/articles/Templates/5242027

questions and their corresponding SPARQLs. Our framework is available as an open sourced repository[6], under a GPL 3.0[7] License.

In the future, we aim to explore more techniques to increase the diversity and complexity of the questions so generated. We will also explore machine translation based techniques to further reduce the need of manually correcting the questions.

Acknowledgements This work was partly supported by the grant from the European Union's Horizon 2020 research Europe flag and innovation programme for the projects Big Data Europe (GA no. 644564), HOBBIT (GA no. 688227) and WDAqua (GA no. 642795).

References

1. K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD Conference on Management of Data*, pages 1247–1250, 2008.
2. Q. Cai and A. Yates. Large-scale semantic parsing via schema matching and lexicon extension. In *ACL*, pages 423–433, 2013.
3. M. Dubey, S. Dasgupta, A. Sharma, K. Höffner, and J. Lehmann. Asknow: A framework for natural language query formalization in sparql. In *International Semantic Web Conference*, pages 300–316, 2016.
4. 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. *The Semantic Web*, pages 1–26, 2016.
5. J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. N. Mendes, S. Hellmann, M. Morsey, P. Van Kleef, S. Auer, and C. Bizer. DBpedia–a large-scale, multilingual knowledge base extracted from wikipedia. *The Semantic Web*, pages 167–195, 2015.
6. G. A. Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.
7. P. Trivedi, G. Maheshwari, M. Dubey, and J. Lehmann. Lc-quad: A corpus for complex question answering over knowledge graphs. In *International Semantic Web Conference*. Springer, 2017.
8. C. Unger, L. Bühmann, J. Lehmann, A.-C. N. Ngomo, P. Cimiano, and D. Gerber. Template-based question answering over rdf data. In *Proceedings of the 21st International World Wide Web Conference*, pages 639–648. ACM, 2012.
9. C. Unger, A.-C. N. Ngomo, and E. Cabrio. 6th open challenge on question answering over linked data (qald-6). In *Semantic Web Evaluation Challenge*, pages 171–177, 2016.
10. D. Vrandečić. Wikidata: A new platform for collaborative data collection. In *Proceedings of the 21st International World Wide Web Conference*, pages 1063–1064, 2012.

[6] https://github.com/AskNowQA/LC-QuAD

[7] https://www.gnu.org/licenses/gpl.html