Question Answering on RDF Data based on Grammars Automatically Generated from Lemon Models

Mohammad Fazleh Elahi[1] $^{[0000-0002-8843-9039]}$, Basil Ell[1,2] $^{[0000-0002-8863-3157]}$
Frank Grimm[1] $^{[0000-0002-7045-8055]}$, and Philipp Cimiano[1] $^{[0000-0002-4771-441X]}$

[1] CITEC, Universität Bielefeld, Germany
`{melahi,bell,fgrimm,cimiano}@techfak.uni-bielefeld.de`
`http://www.sc.cit-ec.uni-bielefeld.de/home/`
[2] Department of Informatics, University of Oslo, Norway
`basile@ifi.uio.no`

Abstract. Many question answering (QA) systems over RDF induced from question-query pairs using some machine learning technique suffer from a lack of controllability, making the governance and incremental improvement of the system challenging, not to mention the initial effort of collecting and providing training data. As an alternative, we present a model-based QA approach that uses an ontology lexicon in lemon format and automatically generates a lexicalized grammar used to interpret and parse questions into SPARQL queries. The approach gives maximum control over the QA system to the developer as every lexicon extension increases the coverage of the grammar, and thus of the QA system, in a predictable way. We describe our approach to generating grammars from lemon lexica and show how these grammars generate specific questions that we index to support fast QA performance in a prototype that answers questions with respect to DBpedia.

Keywords: question answering, RDF, grammar generation

1 Introduction

As the amount of structured data on the Web increases, there is an increasing demand for interfaces that simplify the access and browsing of data by end-users. Approaches to QA over RDF data based on machine learning (see [3] for an overview of deep learning methods applied to QALD and [1] for an overview of recent work on natural language interfaces to databases) face however a number of limitations with respect to the governance and maintenance of the QA system. First of all, the provisioning of training data represents a substantial effort and although transfer learning from an existing dataset to another domain can be applied [5], typically one needs at least a small amount of training data from the target domain to obtain decent performance. Most importantly, QA models induced from training data are not controllable in the sense that it is a priori unclear which questions the model will interpret correctly. Further, the impact of adding a single or a few training examples is not predictable in terms of which additional questions the model will be able to cover.

In order to overcome the problems related to machine learning-based approaches to question answering over RDF, we explore a model-based approach to QA in which a developer of the QA system provides a lexicon in lemon format [7] specifying how the vocabulary elements are realized in natural language. The main benefit of the approach is that it is fully controllable in the sense that it can be predicted what the impact of extending the lexicon will have in terms of the questions covered by the system. To realize the system, we build on our previous work showing how question answering grammars can be automatically generated from lemon lexica [2]. Building on earlier results showing that if the questions of the QALD-7[3] dataset are reformulated in terms of questions that the grammar can cover, we can achieve F-Measures of up to 62.5%, in this paper, we present a QA system that builds on the grammar generation functionality described in previous work. As the main contribution, we show that our approach can scale to large numbers of questions and that the performance of the system is practically in real-time from an end user perspective. We apply our approach to DBpedia and describe the implementation of a QA system that can answer more than 1.8 million questions. The system is available at `https://scdemo.techfak.uni-bielefeld.de/quegg/`.

2 Generating Grammars from Ontology Lexica

Our approach automatically generates lexicalized regular grammars from lexical entries in a lemon lexicon [7] for different parts-of-speech and syntactic behaviours. The approach to grammar generation for (relational) nouns (e.g. *'capital of'*), transitive verbs (e.g. *'(to) direct'*), intransitive verbs subcategorizing a prepositional argument (e.g. *'(to) flow through'*), and intersective adjectives (e.g. *'Spanish'*) were described in previous work [2]. For the sake of self-containedness, we describe the generation of grammar rules for (relational) nouns and the newly deployed gradable adjectives (e.g. *'high/higher/highest'*).

The lemon entry for the relational noun *'capital (of)'* states that the entry has a NounPPFrame [4] that corresponds to a copulative construction *'X is*

[3] http://qald.aksw.org/

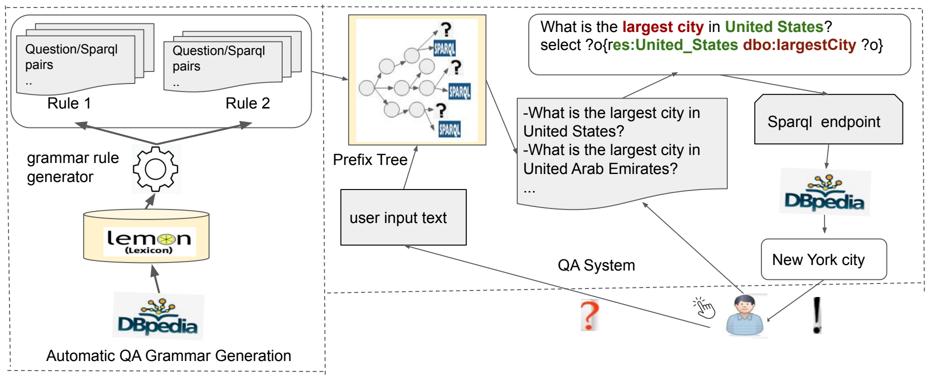
Question/Sparql pairs
..
Rule 1
Question/Sparql pairs
..
Rule 2
grammar rule generator
lemon (Lexicon)
DBpedia
Automatic QA Grammar Generation
Prefix Tree
user input text
What is the largest city in United States?
select ?o{res:United_States dbo:largestCity ?o}
-What is the largest city in United States?
-What is the largest city in United Arab Emirates?
...
Sparql endpoint
DBpedia
New York city
QA System

Fig. 1. The architecture of the QueGG question answering system

the capital of Y' (see [2]). The grammar generation approach for the lemon entry generates the following questions: 1) *'What is the capital of X?'*, 2) *'What was the capital of X'*, 3) *'Which city is the capital of X?'*, 4) *'Which city was the capital of X?'*. The X position can be either a particular country, e.g. *'Germany'*, or a noun phrase, e.g. *'country where German is spoken'*. A second grammar rule for relational nouns not shown in detail here (see [2]) generates noun phrases such as *'the capital of X'*. The code for our grammar generation is available on GitHub.[4]

Gradable adjectives are modelled along the proposal of McCrae et al. [6] and are represented using the lemonOILS[5] ontology. The lexical entry *high* is expressed through the concept `oils:CovariantScalar`, indicating that the adjective is covariant with its bound property `dbo:elevation`. The lexical entry allows our approach covering the following questions: 1) *'How high is X?'* and 2) *'What is the highest X?'*. At position X, the label of individuals of type ArchitecturalStructure can be inserted.

3 System Architecture and Implementation

The core component of the model-based QA system (Figure 1) is the grammar generator, which takes a lemon lexicon as input and automatically creates lexicalized grammar rules, as shown in Section 2, from which pairs of concrete questions and SPARQL queries can be instantiated. The QA component is a web application that maintains a server-side index of question-query pairs, as well as a user-facing web application. The former builds an efficient data structure in order to index the question data for later retrieval. The latter is able to *a)* assist the user in finding the right question through

[4] https://github.com/fazleh2010/question-grammar-generator
[5] `http://lemon-model.net/oils`

Frame type	#Entries	#Grammar rules	#Questions
NounPPFrame	211	424	1060234
TransitiveFrame	32	107	585845
IntransitivePPFrame	52	106	151040
AdjectiveAttributiveFrame	33	130	41425
AdjectiveGradableFrame	8	24	9150
Total	336	791	1847694

Table 1. Frequencies of entries with a certain frame type

auto-complete functionality and *b)* present results given by the SPARQL endpoint in a comprehensive manner.

The index of question-query pairs is a server-side prefix tree built from pre-generated questions. While initial inserts are $\mathcal{O}(n)$ expensive, the structure allows very quick lookups. The tree is populated with lower-cased character sequences of questions. Costly tree maintenance is alleviated indexing content in stages: an initial bulk import and subsequent updates. All application launches, as well as new data insertions, then rely on a previously stored state. All input in the application's query field is periodically pushed to the server, where the tree is then queried for question nodes matching the (lower-cased) input in order to generate auto-complete suggestions. Incomplete questions yield a number of suggestions by means of a breadth-first search limited to a maximum depth of five levels. This produces the most relevant completion paths for the given query. If the maximum number of suggestions was not reached by this search, a second one adds specific questions to the list. When a user reaches a specific question or enters enough information to promote an answerable leaf node to the top of the suggestion list, the system attempts to resolve it. The SPARQL query associated with the active question is sent to the endpoint and various metadata is rendered alongside the answer.

We apply our system to the DBpedia dataset (Release 2016-10; core, links, and English core-i18n) using 336 manually created lexical entries; spreadsheets available at `https://scdemo.techfak.uni-bielefeld.de/quegg-resources/`. Every row added to these spreadsheets increases the coverage of the grammar and generates tens of thousands of new questions. Table 1 shows the number of grammar rules and questions generated for each syntactic type. Altogether, the approach generates 791 grammar rules and about 1.8 million questions. The source code can be obtained via GitHub.[6]. The user-based evaluation of the system involved 161 students (University of Bielefeld) that were asked to enter 5 questions[7] given in German into the

[6] `https://github.com/ag-sc/QueGG-web`
[7] https://forms.gle/B5cjuX5rncxHi1Bx6

English-language QA interface. We evaluate two performance indicators: *a)* Effectiveness: the accuracy and completeness with which participants can ask a question to the system and *b)* Answer Satisfaction: whether participants find the returned answers acceptable.The results show that the tool is intuitively usable, achieving effectiveness and satisfaction rates between 71%–99% and 46%–95%, respectively. The average SUS (System Usability Scale) score obtained is 62.06, which indicated room for improvement.

4 Conclusions

We presented an approach to developing QA systems over RDF datasets that relies on the automatic generation of grammars from corresponding lemon lexica that describe how elements of the dataset are verbalized in natural language. In contrast to machine learning based approaches that induce a model from question-query pairs, our approach is declarative in that the developer of the system defines questions that can be covered by the system by providing a lemon lexicon. The approach is controllable since the introduction of a lexical entry increases the question coverage in a fully predictable way. We have described how an efficient QA system can be implemented on the basis of the automatically generated grammars by indexing the questions and queries using a prefix tree. Our proof-of-concept implementation over DBpedia covers 1.8 million questions generated from 336 lemon entries. In future work we intend to start a community project where the community can contribute both to the extension of the lexicon and the set of grammar rules but also to adapt the grammar generation to other languages.

References

1. Affolter, K., Stockinger, K., Bernstein, A.: A comparative survey of recent natural language interfaces for databases. VLDB Journal **28**, 793–819 (2019)
2. Benz, V., Cimiano, P., Elahi, M.F., Ell, B.: Generating Grammars from lemon lexica for Questions Answering over Linked Data: a Preliminary Analysis. In: NLIWOD workshop at ISWC. vol. 2722, pp. 40–55. CEUR-WS.org (2020)
3. Chakraborty, N., Lukovnikov, D., Maheshwari, G., Trivedi, P., Lehmann, J., Fischer, A.: Introduction to Neural Network based Approaches for Question Answering over Knowledge Graphs. CoRR **abs/1907.09361** (2019)
4. Cimiano, P., Buitelaar, P., McCrae, J.P., Sintek, M.: LexInfo: A declarative model for the lexicon-ontology interface. JWS **9**(1), 29–51 (2011)
5. Maheshwari, G., Trivedi, P., Lukovnikov, D., Chakraborty, N., Fischer, A., Lehmann, J.: Learning to Rank Query Graphs for Complex Question Answering over Knowledge Graphs. In: ISWC Conference. pp. 487–504 (2019)
6. McCrae, J.P., Quattri, F., Unger, C., Cimiano, P.: Modelling the Semantics of Adjectives in the Ontology-Lexicon Interface. In: CogALex Workshop (2014)
7. McCrae, J.P., Spohr, D., Cimiano, P.: Linking lexical resources and ontologies on the semantic web with lemon. In: ESWC Conference. pp. 245–259 (2011)