GQA: Grammatical Question Answering for RDF Data

Elizaveta Zimina, Jyrki Nummenmaa, Kalervo Järvelin,
Jaakko Peltonen, Kostas Stefanidis, and Heikki Hyyrö

University of Tampere, Finland
`firstname.lastname@uta.fi`

Abstract. Nowadays, we observe a rapid increase in the volume of RDF knowledge bases (KBs) and a need for functionalities that will help users to access them in natural language without knowing the features of the KBs and structured query languages, such as SPARQL. This paper introduces Grammatical Question Answering (GQA), a system for answering questions in the English language over DBpedia, which involves parsing of questions by means of Grammatical Framework and further analysis of grammar components. We built an abstract conceptual grammar and a concrete English grammar, so that the system can handle complex syntactic constructions in the focus of the SQA2018 challenge.

Keywords: Grammatical Framework · DBpedia · question answering · SQA.

1 Introduction

In this work, we present the Grammatical Question Answering (GQA) system for answering questions in the English language over DBpedia 2016-04 [3] and trained on the SQA2018 training set [8]. The GQA system's key technology is Grammatical Framework (GF) [6], which provides conceptual parsing of questions and extraction of the necessary components that can be matched with the KB schema and contents to formulate SPARQL [7] queries corresponding to the English questions. GF has previously been applied in several domains, like biomedical linked data [4].

2 GQA Structure

The general architecture of GQA is shown in Fig. 1. GQA processes questions in two steps:

- question parsing, which gives all possible parses of the question according to the GQA grammar, and selection of the best parse,
- interpreting the best parse by "unwrapping" its elements and, starting from the innermost element, making requests to DBpedia, finding the corresponding values and finally the answer(s).

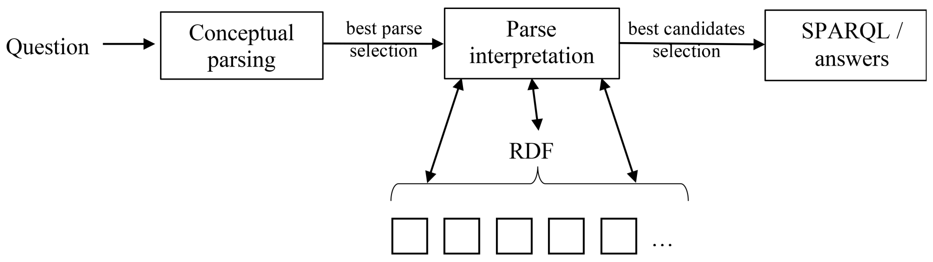

Fig. 1. General Architecture of GQA

Grammatical Framework (GF) [6] is used to parse questions according to an application-specific grammar that we created, which utilises the existing GF resource grammar for the English language. The parse interpretation is realised by means of Python and the index is built for the relevant DBpedia databases using Apache Lucene [1].

3 Grammatical Framework

Grammatical Framework (GF) [6] is a functional programming language and a a categorial grammar formalism, which provides facilities for building grammars for various purposes and using them for string parsing or string generation. A GF programme comprises:

- an abstract syntax, which defines the concepts and their relations within abstract syntax trees,
- a concrete grammar for a certain language (or several grammars for several languages), which specifies the way of mapping abstract syntax trees on to strings and vice versa.

The process of extracting trees from natural language strings is called parsing, and the reverse, linearisation, is used for generation of strings from parse trees.

Our conceptual grammar is organised in a way that it reflects the key elements of questions, so that they can be attributed to certain elements of the KB. At the same time, we employ the existing GF Resource Grammar Library (RGL) [2], a collection of natural language grammars implemented in GF, which makes it easier to handle all possible morphological forms of words.

4 The GQA Grammar

Below, we briefly describe the main conceptual categories that GQA uses.

Simple entity. So-called simple entities in the GQA grammar are usually proper names (*Is **Robert Boyle** a chemist?*), but also can be common names (*Is **lion** an animal?*). Entities should be matched with corresponding links in DBpedia, starting with *http://dbpedia.org/resource/*.

Simple entities are represented in the grammar as sets of strings, that is, they are not covered by the grammar explicitly, but the parser can detect them relying on their position in sentences.

Property. All property labels of DBpedia (almost 17,000) are coded in the GQA grammar. Many of these properties are irrelevant for question answering due to DBpedia's noisiness (e.g. *coachyear6start* or *Mar record low C*), but this corpus contains also useful properties (e.g. *author* or *nationality*), and its size is still not too large to impede the parsing process.

Properties with identical meaning (e.g. *http://dbpedia.org/ontology/parent, http://dbpedia.org/property/parent, http://dbpedia.org/property/parents*) are gathered in a separate database called *Functions*. This allows us to check at the parse analysis step whether an entity contains any of them.

Verb Phrase (VPChunk). Verb phrases usually contain references to DBpedia properties. For example, the predicate in the question *Who owns the ship Victoria?* is attributed to the property *owner*. Our system learns all verb formulations and stores their mappings to certain properties in the *Functions* database.

Verb phrases in the GQA grammar are recognised in all morphological forms, due to the verb paradigms inherited from the RGL grammar. Thus, *own, owns, owned*, etc. will be recognised by the grammar under the rule *own_ V2*, so that we can focus on the verb's semantics rather than its grammatical form.

Some verbs can have different readings. Thus, in the training set the verb *paint* can refer to the properties *creator, artist, author, illustrator,* and *painter.*

Class. *Class* in the GQA grammar corresponds to an ontology class in DBpedia, i.e. a generic term that an entity belongs to (e.g. *Which **comic characters** are painted by Bill Finger?*). *Class* acts as a noun phrase that can be inflected for number and case, since it can be used in questions in various morphological forms (e.g. *What **company's** leader is Edwin Catmull? How many **companies** are located in Toronto?*)

Relative Clause (RelCl). Relative clauses in the GQA grammar can have some regular patterns (e.g. *that + verb phrase*: *Who owns the company **that made the Edsel Villager**?*), or unique ("idiomatic") structures (*What is the television show **whose vocals are performed by** April Stewart and Mona Marshall?*). Attribution of idiomatic clauses to properties is realised through the *Functions* database.

Complex Entity. The functions of simple entities can be performed by more complex structures:

– a property of an entity (*PropOfEnt*), e.g. *Name the mascot of **the military branch of William Harper**.*

- a class with a relative clause (*What is the alma mater of* ***the scientist who is known for rational analysis****?*)
- two entities as homogeneous parts of the sentence (*Which team did* ***Bill Murphy*** *and* ***Jean Segura*** *play for?*)

Question (Q). Questions are categorised by means of rules (currently about 40) constructing the topmost category *Q*, for example:

Who discovered Callisto?
`WhoVP : VPChunk -> Q`
What are the schools whose city is Reading, Berkshire?
`WhatIsClassRelCl : Class -> RelCl -> Q`
Which shareholder of Dagenham wind turbines is also the parent company of the Ford Falcon Cobra?
`WhatPropOfEntIsPropOfEnt : PropOfEnt -> PropOfEnt -> Q`

Similarly to relative clauses, questions can be "idiomatic" (***In which mountain range does*** *the Rochers De Naye* ***lie****?*)

5 Parse Analysis and Answer Extraction

One and the same question can have several readings according to the GQA grammar, thus we need to select the best parse. Largely, this is reduced to selecting a parse with the shortest number of strings (that are regarded by the parser as entities), so that the question tokens are categorised as much as possible.

After the best parse is obtained, the system detects the top function and traverses its arguments down to the bottom nodes of the parse tree, after which it starts making queries to DBpedia to reveal which values are relevant for the current parse. For example, the best parse of the question *What are the awards won by the founder of Walt Disney Records?* looks like:

```
WhatIsX (Class_to_Ent (ClassRelCl_to_Class (ArtAdjClassChunk
(Class_Nom_Chunk Award_CNClass)) (WhoVP_relcl (VPSlash_to_VPChunk
(BeDoneBy_VPslash (DoneBy_PP win_V2)) (NPPropofEntChunk_to_Entity
(MkCNPropOfEnt (ArtAdjPropChunk (Prop_Nom_Chunk founder_O))
(threeWordEnt "Walt" "Disney" "Records"))))))).
```

Due to space limitations, we omit the details of parse here, and outline the general flow of analysis. The system reaches the downmost entity *Walt Disney Records* and looks for its link in the DBpedia index. The *founder* property in our *Functions* database has four alternative readings: *http://dbpedia.org/ontology/founder, http://dbpedia.org/property/founder, http://dbpedia.org/ontology/foundedBy* and *http://dbpedia.org/property/founded*. The entity link contains the third property, so we collect its values (*http://dbpedia.org/resource/Roy_ O._ Disney* and *http://dbpedia.org/resource/Walt_ Disney*). The verb function *win_ V2* refers us to the properties *http://dbpedia.org/ontology/award, http://dbpedia.org/property/awards* and *http://dbpedia.org/property/award* and, matching them with the obtained values, we see that only the entity *Walt Disney* has the property *http://dbpedia.org/ontology/award*. We check that its values (*Academy Awards, Emmy Award,* and *Golden Globe Cecil B. DeMille Award*) have the class *award* and return them as answers.

6 Evaluation

Writing the grammar and its interpretation module is an ongoing process. Currently the system can correctly answer 522 questions out of 5000. On the average it takes 0.57 sec per question using a machine with 64GB of RAM. Except regular sentences, we need to add flexibility to our rules, so that they would accept questions with non-standard word order (*British people have edited which movies?*) and some typical typos. Moreover, we are planning to add an approximate string matching module for better entity resolution (MSX Basics → MSX BASIC, Aston Villa 2000-02 season → 2000–01 Aston Villa F.C. season, etc.).

7 Conclusion

Compared to the QALD challenges of the previous years [5], the SQA2108 task seems to have a greater emphasis on resolution of complex syntactic structures. GQA offers new ways for managing syntactically rich questions and converts them into conceptual parses, which then can be mapped on to DBpedia components. We believe that even though the current system answers a small part of the training set questions, it will be easier to cover new question types in future, since the main components of the system have been already developed.

References

1. Apache Lucene, http://lucene.apache.org/. Last accessed 29 Mar 2018.
2. Bringert, B., Hallgren, T., Ranta, A.: GF Resource Grammar Library: Synopsis, http://www.grammaticalframework.org/lib/doc/synopsis.html. Last accessed 29 Mar 2018.
3. DBpedia version 2016-04, http://wiki.dbpedia.org/dbpedia-version-2016-04. Last accessed 29 Mar 2018.
4. Marginean, A.: Question answering over biomedical linked data with Grammatical Framework. Semantic Web, 8(4):565–580, 2017.
5. Question Answering over Linked Data (QALD), https://qald.sebastianwalter.org/. Last accessed 29 Mar 2018.
6. Ranta, A.: Grammatical Framework: Programming with Multilingual Grammars. CSLI Publications, Stanford (2011).
7. SPARQL 1.1 Overview – W3C, http://www.w3.org/TR/sparql11-overview/. Last accessed 29 Mar 2018.
8. Trivedi, P., Maheshwari, G., Dubey, M., Lehmann, J.: LC-QuAD: A Corpus for Complex Question Answering over Knowledge Graphs. In ISWC 2017.