NL interface to KG

Ritik, Rashi

1. Introduction

1.1 Abstract

Knowledge graphs (KGs) are essential assets for search, analytics, and recommendations. However, querying a KG to explore entities and discover facts is stressful and tedious, even for users with skills in SPARQL. Hence, one requires expert knowledge of the query language's syntax and semantics. We will show an NL query interface, which is an easy way of accessing a knowledge graph by any user who typically may not have any knowledge about query languages to query natural language questions to the DBpedia endpoint.

1.2 Motivation

An NL query interface allows users to interact with the computer using a human language, such as English or Chinese, instead of using a computer language. For example, when confronted with a question of the form 'which U.S. state has the highest income tax?', conventional search engines ignore the problem and instead search on the keywords 'state,' 'income' and 'tax.' On the other hand, natural-language search attempts to use natural-language processing to understand the nature of the question and then to search and return a subset of the web that contains the answer to the problem.

1.3 Objective

- Map natural language to an understandable, logical form i.e. lambda DCS.
- Define basic grammar rules to map complex structures exploiting compositionality.
- Query the DBPedia endpoint to obtain the answer to a question.

2. Framework

2.1 SEMPRE

A semantic parser maps natural language utterances into an intermediate logical form, which is "executed" to produce a useful denotation for some tasks.

A question answering task:

- Utterance: *Where was Obama born?*
- Logical form: (place_of_birth barack_obama)

- Denotation: Honolulu

By parsing utterances into logical forms, we obtain a rich representation that enables much deeper, context-aware understanding beyond the words. With the rise of natural language interfaces, semantic parsers are becoming increasingly more powerful and useful.

SEMPRE is a toolkit that makes it easy to develop semantic parsers for new tasks. The primary paradigm is to learn a feature-rich discriminative semantic parser from a set of utterance-denotation pairs. One can also quickly prototype rule-based systems, learn from other forms of supervision, and combine any of the above. **Although SEMPRE is designed primarily for the Freebase KG, we have made some modifications to generate queries tailored to DBpedia.**

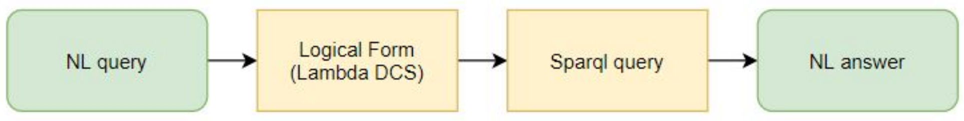

Fig.1 SEMPRE Architechure

2.2 EARL

Although some Semantic Question Answering (SQA) systems have achieved excellent performance on simple questions, i.e., those questions which can be answered by linking to at most one relation and most one entity in the KG, recently, the focus has shifted towards complex questions, comprising of multiple entities and relationships. The main challenges faced in SQA are (i) entity identification and linking, (ii) relationship linking.

To achieve this, we used EARL (Entity and Relation Linker), a system for jointly linking entities and relations in a question to a knowledge graph.EARL uses the knowledge graph to disambiguate entity and relations together: It obtains the context for entity disambiguation by observing the relations surrounding the entity. Similarly, it obtains the context for relation disambiguation by looking at the surrounding entities. The system supports multiple entities and relations occurring in complex questions.

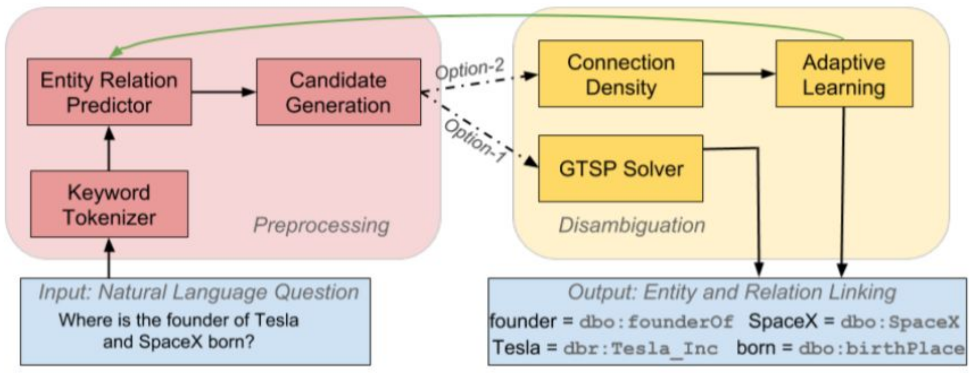
Entity Relation Predictor
Candidate Generation
Keyword Tokenizer
Preprocessing
Option-2
Option-1
Connection Density
Adaptive Learning
GTSP Solver
Disambiguation
Input: Natural Language Question
Where is the founder of Tesla and SpaceX born?
Output: Entity and Relation Linking
founder = dbo:founderOf SpaceX = dbo:SpaceX
Tesla = dbr:Tesla_Inc born = dbo:birthPlace

Fig.2 EARL Architecture

3. Approach

3.1 Lambda DCS

SPARQL is the standard language for querying graph databases, but it will be convenient to use a language more tailored for semantic parsing. We will use lambda DCS, which is based on a mix between lambda calculus, description logic, and dependency-based compositional semantics (DCS). Here is an example,

- Utterance: “people who have lived in Seattle”
- Logical form (lambda calculus): $\lambda x.\exists e.\text{PlacesLived}(x, e) \land \text{Location}(e, \text{Seattle})$
- Logical form (lambda DCS): PlacesLived.Location.Seattle

Here are the following types of logical forms in lambda DCS:

1. Primitive (e.g., dbr:California): denotes a set containing that single entity.
2. Intersection (and |u1| |u2|): denotes the intersection of the sets denoted by unary logical forms u1 and u2.
3. Join (|b| |u|): denotes the set of x which are connected to some y via a binary b and y is in the set denoted by unary u.
4. Count (count |u|): denotes the set containing the cardinality of the set denoted by u.
5. Lambda abstraction (lambda (var |v|) |u|): produces a binary (x,y) where x is in the set denoted by u and y is the value taken on by variable v.

3.2 Parsing and Grammer

We will now focus on parsing the natural language utterances into the above logical forms. The core challenge in parsing is at the lexical level: mapping natural language phrases to logical predicates. It is useful to distinguish between two types of lexical items:

- Entities (e.g., dbr:Barack_Obama): There are generally a huge number of entities (DBpedia has tens of millions). Often, string matching gets you part of the way there, but there is often quite a bit of ambiguity (Obama is also a city in Japan).

- Non-entities (e.g., dbo:birthPlace), which include unary and binary predicates: There are fewer of these, but string matching is unlikely to get you very far.

Therefore, we define some grammar rules, which specify how to combine logical forms to build more complex ones in a manner that is guided by the natural language. For simple phrases, we could define simple grammar rules like,

(rule $Entity (Tom Cruise) (ConstantFn dbr:Tom_Cruise))

but grammars are supposed to be small, so this approach does not scale, so we are not going to do this. Hence instead, we create a lexicon, which is a mapping from words to predicates, with entries like this:

1) {"lexeme": "Tom Cruise", "formula": "dbr:Tom_Cruise"}
2) {"lexeme": "films", "formula": "(rdf:type dbo:Film)"}
3) {"lexeme": "starring", "formula": "dbo:starring"}

Then we can add the following rules:

1) (rule $Unary ($PHRASE) (SimpleLexiconFn))
2) (rule $Binary ($PHRASE) (SimpleLexiconFn))
3) (rule $Set ($Unary) (IdentityFn))
4) (rule $Set ($Unary $Set) (MergeFn and))
5) (rule $Set ($Binary $Set) (JoinFn forward))
6) (rule $ROOT ($Set) (IdentityFn))

The SimpleLexiconFn looks up the phrase and returns all formulas that have the given type. MergeFn takes the two (unary) logical forms |u| and |v| (in this case, coming from $Unary and $Set), and forms the intersection logical form (and |u| |v|). JoinFn takes two logical forms (one binary and one unary) and returns the logical form (|b| |v|).

On combining the grammar and the lexicon, we can now parse the following utterance,

Input: *Which are the films starring Tom Cruise?*

Logical Form: *(and (rdf:type dbo:Film) (dbo:starring dbr:Tom_Cruise))*

3.3 Implementation

- Removed the stopwords from the set of questions, such as “is”, “in”, “for”, “where”, “when”, “to”, “at” etc.
- Used the EARL framework to generate the lexicon file which had had mapping from phrases to DBpedia surface forms.
- Created a simple yet efficient set of grammar rules which could parse complex query structures of various compositions
- Ran the questions through the SEMPRE framework to generate the logical forms and their corresponding SPARQL query.
- Used the DBpedia endpoint to get the answers for the SPARQL queries.

4. Results and Discussion

We used the questions from the QALD dataset. There were 148 questions in total where there were paraphrases as well. After generating the SPARQL queries, we got answers for 46 questions, hence the efficiency of the parser was **31.08%**. The queries which couldn't retrieve any answer didn't possess the same structure as found in the DBpedia KG.

Among the 46 questions, 23 questions were answered correctly, 7 questions were partially correct and 16 questions were answered incorrectly. Hence the accuracy of our NL interface is **15.54%**. Here are the examples of the types of questions along with their answers,

- Input: Where is Fort Knox located?

  Output: Kentucky (Correct)

- Input: Where is Fort Knox situated?

  Output: Kentucky (Correct)

- Input: Who was influenced by Socrates?

  Output: Paramenidies, Anaxagoras, Prodicus, etc. (Correct)

- Input: Did Che Guevara have a son or daughter or both?

  Output: Aleida Guevara (Partially Correct)

- Input: What is the biggest stadium in Spain?

  Output: ACD San Marcial, AD Adra, etc. (Incorrect)

Hence, it performs well in answering **single/multi fact questions** primarily and even handles the **paraphrases** giving the same answers, but is unable to process questions with deeper meanings that couldn't be captured in the grammar. Also, the information extraction might sometime yield irrelevant surface forms, which produces incorrect answers.

For example, in "*Where did Nikos Kazantzakis die?"* die is mapped to dbo:deathCause, but actually it should have been mapped to dbo:deathPlace. Also, in "*What is the biggest stadium in Spain?"* the word biggest couldn't be captured and an incorrect expression is formed. And lastly, in "*Did Che Guevara have a son or daughter or both?"* the type of question is boolean but it is answered as a single fact question which is not handled by the grammar.

5. Conclusion

In this project, our main aim was to generate a complete pipeline that would take a natural language question as input and answer it from the DBpedia KG. We used the framework of SEMPRE to generate comprehensible intermediate forms to query the KG. But the challenge here was to use SEMPRE for DBpedia because it has been designed to work on Freebase only. Also, EARL allowed generating the lexicon file, which was much needed to map phrases to their surface forms, with very high efficiency. In the end, we generated a very simple set of grammar rules to parse complex structures, which also showed promise. The results obtained on the QALD dataset were good given the duration of this project but they still have scope for colossal improvement if the grammar rules are improved and the relation linking is done better.

6. References

[1] Dubey, Mohnish & Banerjee, Debayan & Chaudhuri, Debanjan & Lehmann, Jens. (2018). EARL: Joint Entity and Relation Linking for Question Answering over Knowledge Graphs.

[2] Berant, J. & Chou, A. & Frostig, R. & Liang, P.. (2013). Semantic parsing on freebase from question-answer pairs. EMNLP 2013 - 2013 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference. 1533-1544.