Deep Semantic Parsing Method to Capture the Structure of Multi-Relation Questions over Linked Data

Wafa Nouar
University of Constantine 2 - Abdelhamid Mehri
LIRE Laboratory
Constantine, Algeria
wafa.nouar@univ-constantine2.dz

Zizette Boufaida
University of Constantine 2 - Abdelhamid Mehri
LIRE Laboratory
Constantine, Algeria
zizette.boufaida@univ-constantine2.dz

Abstract

Some Question Answering Systems over Linked Data have achieved good performance on simple questions, i.e. questions which can be answered by linking to at most one relation and at most one entity in the Knowledge Graphs. Recently, the focus has shifted towards complex questions, comprising multiple entities and relations. In this paper, we present a question answering system that takes as input a multi-relation question formulated in English and generates an equivalent SPARQL SELECT query. The proposed system is equipped with a novel method that capture the complete structure of multi-relation questions by using Semantic Parsing techniques which generates formal meaning representations in the style of Discourse Representation Theory. This method offers a deep understanding of complex questions.

1. Introduction

With the rapid development of the Web of Data, there are many RDF datasets published as Linked Data [Lehmann2015]. The term Linked Data refers to a set of best practices for publishing and connecting structured data on the Web. These practices were defined by Tim Berners-Lee in 2006 [Bizer2009]. This Linked Data usually has complex structures and are highly heterogeneous. As a result, several gaps between the users and the Linked Data are emerged. Moreover, the users, even expert programmers, need a lot of practices to handle standard structured query languages like SPARQL [He2013]. Thus, developing user-friendly interface for accessing this Linked Data become increasing important. Question Answering Systems over Linked Data is aimed at eliminating those gaps, attempts to allow users to express arbitrarily complex information needs in an intuitive fashion and, at least in their own language, to find the corresponding answers in knowledge Base (KB) or Linked Data and to present the answers in an appropriate form [Unger2015].

Currently most proposed question answering systems (QAS) are restricted to simple questions with a single relation and entity in which the task of query construction is straightforward. As a typical example, the question "*Who created the character Harry Potter*" can be answered with the single fact (*HarryPotter, CharacterCreatedBy, J.K.Rowling).* In comparison, reasoning over multiple fact triples is required to answer multi-relation questions such as "*Name a soccer player who plays at forward position at the club Borussia Dortmund.*" where more than one entity and relation are mentioned. The effort over complex questions (the question that contains one relations) is not satisfactory.

The QAS must determine the overall logical structure of the input natural language (NL) question. Problems arise when several facts have to be found out, connected and then combined respectively the resulting query has to obey certain restrictions. To do this, current QAS need to rewrite the complex questions when it is unable to fully capture with templates, i.e. a set of predefined queries with some slots that have to be filled, but this is not sufficient for longer, more complex sentences. For complex questions, sophisticated approaches are required to capture the structure of the underlying query. If the generated structured query is incorrect, the QAS is likely to find false or incomplete answers. Hence, generating a structured query for the input question is a crucial step.

In summary, our contribution in this paper is a QAS equipped with a semantic parsing method to capture the structure of multi-relation questions, i.e. complex questions that require multiple knowledge graphs (KG) predicates to obtain an answer. For example, the question "*Which were the alma maters of the PR managers of Hillary Clinton*?" is answered by a chain of KG predicates.

The semantic parse used in this method provides a deeper understanding of the question. We opt for Discourse Representation Theory (DRT), a widely accepted sophisticated formal theory of NL meaning dealing with a large variety of semantic phenomena. Advantageously, it supports meaning representations for entire texts rather than isolated sentences. DRT can be translated into First Order Logic (FOL), and later converted FOL queries into executable queries, including SPARQL. This allows us to choose the most appropriate query language technology for a given use case.

The remainder of this paper is organized as follows: we introduce some useful notions in section 2. In section 3, we discuss some work related to our method. Section 4 details our proposition. In section 5, we apply the proposed method on a case study. Finally, section 6 concludes the paper and suggests some future work.

2. Discourse Representation Theory

Semantic parsing is the task of mapping NL text to formal representations or abstractions of its meaning. A variety of meaning representations have been adopted over the years. DRT [Kamp2005] is a popular theory of meaning representation originally designed to cope with anaphoric pronouns and temporal relations and other contextually sensitive linguistic phenomena. A single semantic formalism, DRT, embraces all these phenomena by taking meaning representations of texts rather than sentences which in turn can be translated into FOL. Semantic representations in DRT are specified in terms of a language based on boxlike structures called DRSs (Discourse Representation Structure). The non- terminal <ref> denotes a discourse referent, and <symn> an n-place predicate symbol.

<expe> ::= <ref>
<expt> ::= <drs>

<drs>: =

<ref>∗
<condition>∗

<condition> ::= <basic> |
<complex>
<basic> ::= <sym1>(<expe>) |
<sym2>(<expe>,<expe>) |
<named>(<expe>,<nam>,<sort>)
<complex> ::= ¬<expt> |
<expt> ⇒ <expt>
<expt> ∨ <expt> |
<ref> : <expt>

DRSs are structures comprising two parts: **i)** a set of discourse referents; and **ii)** a set of conditions constraining the interpretation of the discourse referents i.e. information that has accumulated on these discourse referents. Conditions are properties of discourse referents, express relations between them.

- **Accessibility in DRT**

The accessibility constraint imposes restrictions on the interpretation of anaphoric expression. In linguistics, an anaphor is the phenomenon of one linguistic expression (typically a pronoun) referring to another linguistic expression in the same discourse to avoid repetition. Anaphoric pronouns are represented as free variables linked to appropriate antecedent variables.

Accessibility is a geometric concept, defined in terms of the ways DRSs are nested into each other. A DRS B_1 is accessible from DRS B_2 when B_1 equals, or when B_1 subordinates B_2 [Kamp2005].

- ***Subordination***

A DRS B_1 subordinates B_2 if:

- B_1 immediately subordinates B_2.
- There is a DRS B such that B_1 subordinates B and B subordinates B_2.

B1 immediately subordinates B2 if:

- B1 contains a condition¬B2.
- B1 contains a condition B2 ∨ B or B ∨ B2.
- B1 contains a condition B2⇒ B.
- B1⇒B2 is a condition in some DRS B.

3. Related Work

In this section we present an overview on some QAS dealing with multi-relation questions. We focus on the strategies used to capture the structure of the question.

Many template-driven approaches have been proposed. An early contribution was proposed in [Adolphs2011] where the authors presented YAGO-QA which is a QAS that allows nested queries when the subquery has already been answered. For example, *"Who is the governor of the state of New York?*" after "*What is the state of New York?*". This QAS also answers complex questions where the queried entities are not directly related to the entities given in the question but are linked to them through a chain of relations. For instance, rather than asking "*Who invented x-bar theory*?" (A: "Noam Chomsky") and then using the result of that question in a follow-up question "*Where does Noam Chomsky work?*" ("MIT"), YAGO-QA can directly find the result with the question: "*Where does the person work who invented x-bar theory?*". Query templates are instantiated by filling all named entities slots with the highest ranked referent and then issued to the KB. However, not all questions can be treated using templates. Moreover, fuzzily mapping user questions against a set of paraphrases is not sufficient for dealing with the productive use of language.

Template-based approaches range from handcrafted or dynamically built templates. In [Abujabal2017] the authors presented a QAS that handle compositional questions, for example "*Who won a Nobel Prize in Physics and was born in Bavaria?*". The QAS relies on question-query templates, previously learned by distant supervision, which are ranked on several features by a preference function learned via random forest classification. This QAS is able to harness language compositionality for answering multi-relation questions without having any templates for the entire question, but generate and execute candidate SPARQL queries for each sub-question separately and/or use the intersection as the final answer *(Werner Heisenberg, among others)*. However, this creates the challenge of deciding which queries from the different sub-questions fit together. Also, the method has a limited ability to handle complex questions since it highly depends on the syntactic parser and manually defined rewriting rules.

A very recent work was proposed [Zheng2018], where the authors propose a method to understand NL questions by using a large number of binary templates. They present a low-cost approach that can build a huge number of templates automatically. They design a systematic technique to accelerate the question decomposition and handle the two-level ambiguities effectively (i.e., entity-level ambiguity and structure-level ambiguity) by considering the query semantics.

Template-free approaches try to build SPARQL queries based on the given syntactic structure of the input question. Intui2 [Dima2013] and Intuit3 [Dima2014] accepts as input a NL question and constructs its interpretation using syntactic and semantic cues in the question and a target triple store. First, the question is syntactically analyzed and chunked, and the named entities are identified. Then each chunk receives one or more interpretation depending on its type and on additional semantic and syntactic information available for that chunk. The interpretation of the question is then constructed by combining the interpretations assigned to each chunk, based on a set of combination rules that are attached to each type of interpretation. If necessary, some additional variables between the identified segments are added, for example if one relation is followed by another relation. However, the two QASs make the implicit assumption that it is possible to deduce the structure of the SPARQL query from the structure of the question without knowing how the knowledge is encoded into the KB. Moreover, approaches followed this path require additional effort of making sure to cover every possible basic graph pattern. Thus, only a few QAS tackle this approach so far.

GETARUNS [Tripodi2013] first creates a logical form out of a query which consists of a focus, a predicate and arguments. The focus element identifies the expected answer type. For example, the focus of "*Who is the major of New York?*" is "person", the predicate "be" and the arguments "major of New York". If no focus element is detected, a yes/no question is assumed. In the second step, the logical form is converted to a SPARQL query by mapping elements to resources via label matching. The resulting triple patterns are then split up again as properties are referenced by unions over both possible directions, as in ({?x ?p ?o} UNION {?o ?p ?x}) because the direction is not known beforehand. Additionally, there are filters to handle additional restrictions which cannot be expressed in a SPARQL query, such as "*Who has been the 5th president of the USA*".

In [Shekarpour2015], the QAS called SINA use Hidden Markov model to address the ambiguity that arises in the mapping phase. An advantage of this technique is that it is not necessary to know the dependency between the different resources but only a set of possible resource. The disadvantages are that this process is computationally complex and that the syntax of the

question is not respected. For example, the QAS will not see any difference between the questions "*Who is the mother of Angela Merkel*?" and "*Angela Merkel is the mother of who?* ».

In the literature we can find other researches that captures the structure of complex questions using machine learning techniques. This allows more flexibility. However, this comes with the price of a training phase. In this paper we demonstrate that in general it is impossible to deduce the form of the SPARQL query knowing only the question. So, we present a method that focuses on both features obtained from the question analysis phase and semantic information from Linked Data resources. Unlike other strategies that use only one feature, in this method the query is constructed starting from the question analysis and using the KB. We choose to use DRT because it has established itself as a well-documented formal theory of meaning witch use DRS that offers suitable level of NL meaning representation. Also, DRT provide a deep parsing of NL and capture complex relations. The intermediate level of DRSs, or semantic representations, is the key to the theory's novelty. DRS represents both context and content. Also, we use FOL to capture NL meaning due to their expressiveness, its well understood formalism, and for the availability of promising tools. In particular logical inference can contribute to ambiguity resolution in discourses, for instance, concerning the resolution of anaphoric expressions. The proposed QAS exploits NL compositionality without using any templates. The main advantage of this approach is that we can directly get a semantic representation.

4. Our proposition

Assuming answering simple WH-questions is possible using a search engine. The proposed QAS translates multi-relation questions to a formal representation. (In this QAS we are limited to factual questions because others, such as why i.e. causal or how i.e. procedural questions require more additional processing). Figure 1 illustrates the QAS architecture.

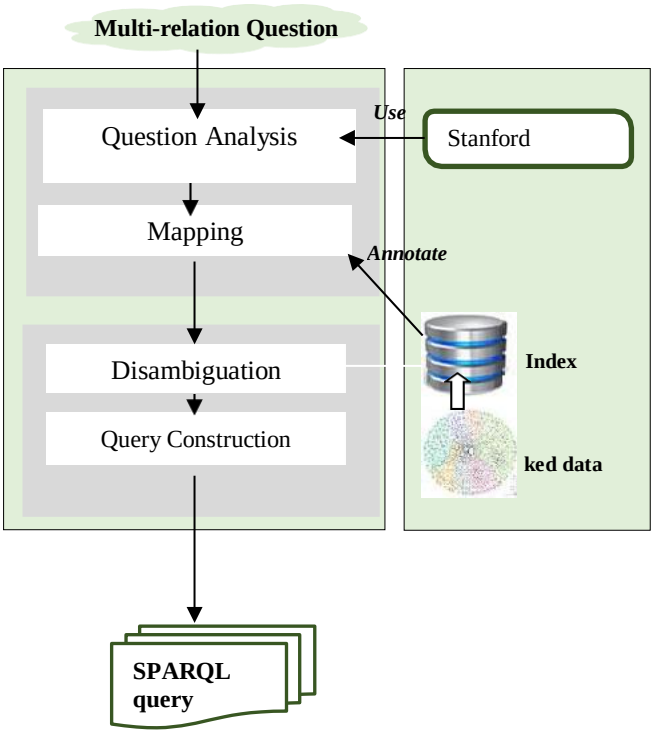


Figure 1: Overall system architecture.

The following sections explain the system phases in more detail. To answer a question in reasonable time, the QAS relies on an index in order to match NL expressions with labels of vocabulary elements.

- **Question analysis phase**: This processing involves a syntactic and a semantic analysis. In this phase we find how the terms in the question relate to each other and make a hypothesis about the correspondence. The right one can be selected afterwards.

- **Mapping phase**: Based on the Linked Data consisting of subject-property-object (SPO) triples, each NL question firstly is transformed into a triple-based representation (Query Triple). Then, the corresponding resources in the KB are mapped for the phrases in the query triples.

- **Disambiguation phase:** In the disambiguation task, we determine which of the resources identified during the phrase mapping task are the right ones. For example, "Apple " cannot refer to a company or a fruit.

- **Query construction phase**: This component creates the final SPARQL query using information provided by above tree steps i.e. and using both question features extracted from question analysis phase and semantic information from KB.

We make use of some reusable question answering components to compose the full QAS pipeline above. We focus on both question analysis and query construction task where we use a semantic parsing method. The process includes four steps:

- Step 1: Syntaxic analysis of the query which involves both segmenting the question using POS tagging and dependency parsing. POS-tagging refers to the process of dividing the sentence into noun, verb, pronoun, adverb, helping-verb, adjective, prepositions, etc. In our case we choose use Stanford CoreNLP1. This open source analysis tool can give the base forms of words, their parts of speech, whether they are names of companies, people, etc., normalize dates, times, and numeric quantities, mark up the structure of sentences in terms of phrases and word dependencies, indicate which noun phrases refer to the same entities.

- **Step 2**: Named Entity Recognition refers to the task of correctly identifying atomic entities in text that fall into predefined categories such as person, location, organization, etc.

- **Step 3:** Linguistic triple generation where the purpose of this linguistic disambiguation is to generate linguistic triples that semantically and syntactically comply with the question posed by the user. It focuses solely on understanding the question posed by the user, without involving any KB concept matching.

- **Step 4:** Semantic parsing of the query couple syntactic rules to semantic composition using DRS construction algorithm proposed by [Kamp2005].

5. Case study

To better understand the sequence of previous steps. Let us consider the question "Who [**was married to**] ***relation 1*** an actor that [**played in**] ***relation 2*** Casablanca? This query contains two relations (was married to, played in).

All NL semantics are concerned with the question how meaning is determined by syntactic form. Thus, every explicit semantic analysis must assume some forms of syntactic structure for the NL expressions with which it is concerned, so as a preliminary to semantics, we need syntax.

Step 1: Syntaxic analysis of the query

- ***POS-tagging***: Who/WP was/VBD married/VBN to/TO an/DT actor/NN that/WDT played/VBN in/IN Casablanca/NNP ?/.

Table 1: Abbreviations used by the Stanford CoreNLP to denote pos tags.

Pos-tagging	**Explication**
WP	Wh-pronoun
VBD	Verb, past tense
VBN	Verb, past participle
DT	Determiner
NN	Noun, singular or mass
WDT	Wh-determiner
NNP	Proper noun, singular

The proposed QAS follows a compositional approach to develop the logical formula corresponding to the question. The main idea of Frege's Principle of Compositionality can be expressed the following way: the interpretation of a complex expression is determined by the interpretation of its constituent expressions and the rules used to combine them. Such an approach requires a kind of syntactic processing by grouping word into larger syntactic units and ordering them as a tree to guide the recursive computation.

[1] http://corenlp.run/

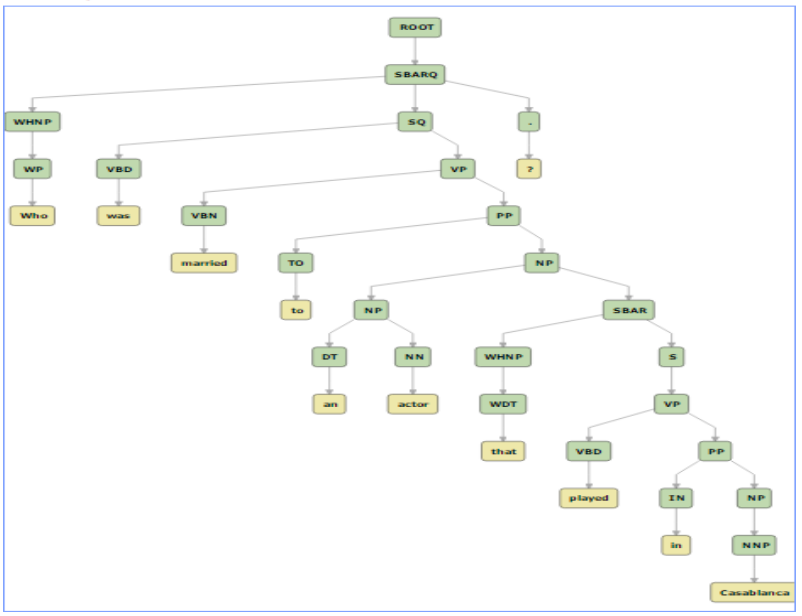


Figure 2: Screenshot of the parse tree for the question.

- ***Dependency parsing***: In such question's dependency parsing is necessary because the question refers to multiple relationships. Dependencies generated by Stanford CoreNLP are binary relation used in all modules of the system to identify the relations between two words of a sentence.

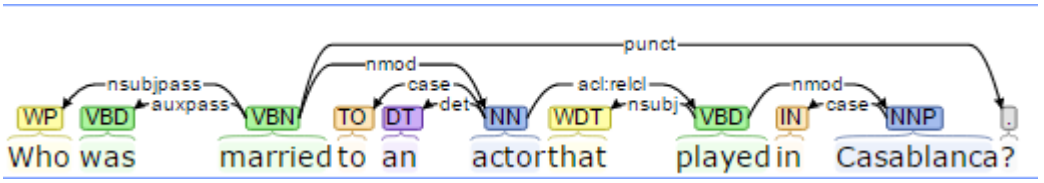


Figure 3: Screenshot of dependency parsing obtained by Stanford CoreNLP.

Table 2: Abbreviations used by the Stanford CoreNLP to denote grammatical relationships

Grammatical relations	**Explication**
nsubjpass	Nominal subject (passive)
auxpass	Auxiliary (passive)
punt	punctuation
nmod	Nominal modifier
case	Case marker
det	Wh-determiner

- **Step 2: Named Entity Recognition**

In this QAS we use ***N-gram strategy***, a common strategy is to try to map n-grams (groups of n words) in the question to entities in the underlying KB. If at least one resource is found, then the corresponding n-gram is considered as a possible named entity. This has the advantage that each named entity in the KB can be recognized. The validated segments are: '*was married to'*, *played in*', '*actor*' and '*Casablanca*'. The named entity is *Casablanca.* We decide already in this step what part of the question corresponds to an instance, relation and class. This way we avoid annotation error.

- **Step 3: Linguistic triples generation**

Linguistic triples are generated based on words in the NL question. Each triple consists of < subject > < predicate > <object > and each NL question may have 1 or more linguistic triples. The generated linguistic triples are mapped to KB triples to find resources that have complete or partial answers to the NL question.

In this step, the corresponding query triples generated for the question are:

1/ ? / Married to / actor.
2/ actor/ played in / Casablanca.

Note that the subject of the second triple 1 is unknown. These triples will serve as semantic information to the query construction task.

- **Step 4: Semantic analysis of the query.**

In the original formulation of DRT [Kamp2005] the construction of DRSs is spelled out in terms of an algorithm based on DRS construction rules. These rules allow to transform a syntactic analysis into a deep semantic structure or a DRS. It successively decomposes syntactic analyses for the individual sentences in a discourse into DRSs in a roughly top-down, left-to-right manner. Applying a given rule consists therefore in replacing the syntactic representation for its corresponding discourse referents and associated conditions. The output of this algorithm is a set of DRSs.

DRS Construction Algorithm

Input :
A discourse D = S*1*,…..,Si; S*i+1*,….., S*n*
the empty DRS K0

Keep repeating for *i* = 1,…..,n:

(i) Add the syntactic analysis [Si] of (the next) sentence Si to the conditions Ki-1; call the DRS Ki*. Go to (ii).

(ii) Input: a set of reducible conditions of Ki*

Keep on applying construction principles to each reducible condition of Ki* until a DRS Ki is obtained that only contains irreducible conditions. Go to (i).

Figure 4 : Standard DRS Construction Algorithm.

Consider the query "***Who was married to an actor (1) that played in Casablanca***?" **(2)**. This query contains two sentences. As a first step the DRS construction algorithm inserts the syntactic analysis of the first sentence in **(1)** into an empty DRS representing an initial empty context. Next, a DRS construction rule for indefinite noun phrases (NPs) matches the relevant part of the parse tree, introduces a ***new*** discourse referent X into the universe of the DRS under construction and adds a condition ***Was married (x, ?)***. The matching part of the tree configuration is replaced by X. Finally, the matching tree is consumed and a condition ***Actor (x)*** is added to the DRS condition. This completes the processing of the first sentence of **(1)**.

X
Was married to (x,?) Actor (x)

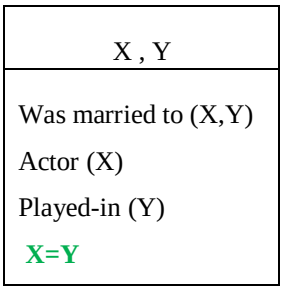
The processing of a piece of discourse is incremental. So, in the next step the top-down construction algorithm inserts the syntactic analysis of the second sentence **(2)**.

Y
Played-in (Y)

The current context is merged with the sentence DRS to yield the new context. The merge conjoins two DRSs into a larger DRS. Semantically the merge is interpreted as dynamic logical conjunction. First a DRS construction rule for pronominal NPs introduces a new discourse referent **Y** into the universe of the DRS under construction. Next it adds a condition **Y = ?** to the set of conditions and replaces the matching part of the tree with **Y**. Informally, **Y =?** can be understood as an instruction

to find a suitable antecedent for the anaphoric pronoun ***that***. Such anaphoric pronouns are subject of ***accessibility*** relation. *(see section 2).*

A DRS B1 is accessible from DRS B2 when B1 equals B2, or when B1 subordinates B2. In the case at hand the anaphoric pronoun ***that*** introduce the subordinate clause ***(played in Casablanca)***. So, the discourse referent X is available and the anaphor is resolved to **Y = X**. Note that in this set-up the anaphoric pronoun ***that*** is resolved as soon as it is processed by the construction algorithm. While processing the second sentence, we establish the links between the pronouns and their intended antecedent. So, we obtain a DRS, which embodies the information conveyed by both sentences of the query.



[*The context after merging*]

- ***Translation from DRS to FOL (Interpreting DRS)***

Once we have built DRS, we translate them into FOL to define the semantics for the DRS language. The standard translation from DRSs into first-order formulas proceeds as follows: each discourse referent is translated as a first-order quantifier, and all DRS-conditions are translated into a conjunctive formula of FOL. Discourse referents usually are translated to existential quantifiers, with the exception of those declared in antecedents of implicational DRS-conditions, that are translated as universal quantifiers. Obviously, negated DRSs are translated as negated formulas, disjunctive DRSs as disjunctive formulas, and implicational DRSs as formulas with material implication. We use the function (.)f° to translate DRSs into first-order formulas

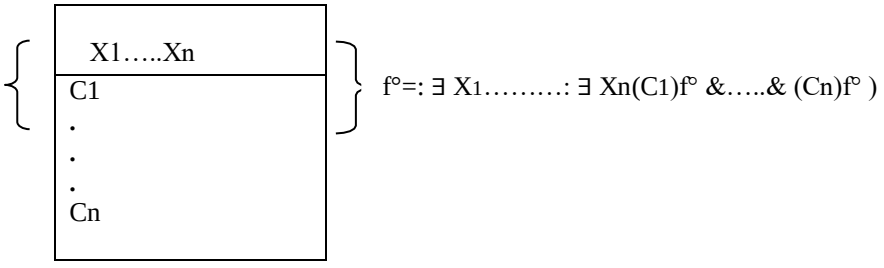


Figure 5: Translating DRSs to FOL

Different translators are developed to further translate the FOL query into SPARQL. For example, the system [Song2015] summarize the process of translating FOL to SPARQL/SQL. First, it parses the FOL representation into a parse tree according to an FOL parser. This FOL parser is implemented with parser development tool. The FOL parser takes a grammar and an FOL representation. Next, then it performs an in-order traversal (with parser development APIs) of the FOL parse tree and translate it to executable queries. While traversing the tree, the system put all the atomic logical conditions and the logical connectors into a stack. When it finishes traversing the entire tree, it pops the conditions out of the stack to build the correct query constraints; predicates in the FOL are also mapped to their corresponding attribute names (SQL) or ontology properties (SPARQL).

In our query example, translation from a NL question to SPARQL query via a FOL representation:

- **NL question**: “Who was married to an actor that played in Casablanca”.
- **FOL Representation**, we can formulate the query as follows: ∃ X. ∃Y: Actor(x) ∧ was-married to (x, y) **∧** played-in-Casablanca (y).

SPARQL is a query language for RDF, which is essentially just binary predicate logic. Together, FOL and RDF form a logical system where RDF provides the non-logical symbols (the vocabulary, the predicates), and FOL provides the logical connectives (logical *and*, or, *-not*), quantifiers, and variables. So, FOL+RDF triples certainly provide the necessary building blocks to write down any SPARQL query.

After adopting formal semantic representation coupled with triples obtained from Linked Data, the resulting SPARQL query (*prefixes are omitted*[2]*)* for the example above is:

```
Select ?x where
{
?x      rdf:type res: Actor.
 ?x     was-married  ?y.
 ?y      rdf : type          dbo:Person.
 ?x      dbp: played-in       res: Casablanca.
 }
```

The KB used by this QAS is ***DBpedia***3 which is based on cross-domain ontology with most concepts representing places, persons, work, species, and organizations. The major advantage of using DBpedia as linking hub is that it contains semantic relations bridging different domains. This way specialized domain-specific datasets linked to DBpedia can be leveraged in cross domain (and cross dataset) queries.

6. Conclusion and future work

Multi-relation Question Answering is a challenging task, due to the requirement of elaborated analysis on questions and reasoning over multiple fact triples in KB. In this paper, we presented a QAS that translates multi-relation questions into SPARQL queries using a semantic parsing method. The proposed method relies on deep semantic representation derived from a syntactic analysis. The key advantage of this method is that the semantic structure of the question is faithfully captured so complex questions containing multiple relations pose no problem. However, in this QAS, input questions have to be complete and well formulated, as ongoing research, we want to enhanced the system to deal with respect to malformed questions. Moreover, we are planning to extend our method to handle more complicated situations where arithmetic operation or commonsense reasoning is needed.

References

[Lehmann2015] 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.

[Bizer2009] C. Bizer, T. Heath, and T. Berners-Lee. Linked data: The story so far. International Journal on Semantic Web and Information Systems (IJSWIS), 5(3):1–22, 2009.

[Unger2015] C. Unger, C. Forascu, V. López, A.-C. N. Ngomo, E. Cabrio, P. Cimiano, S. Walter, C. Forescu, V. Lopez, A.-C. N. Ngomo, E. Cabrio, P. Cimiano, and S. Walter. Question Answering over Linked Data (QALD-5). 2015.

[He2013] S. He, K. Liu, J. Zhao, S. Liu, Y. Chen. Casia@ qald-3: A question answering system over linked data. In: Proceedings of the Question Answering over Linked Data lab (QALD-3) at CLEF 2013.

[Kamp2005] H. Kamp, J.V. Genabith, U. Reyle. Discourse Representation Theory. In Dov M. Gabbay & Franz Guenthner (eds.), Handbook of Philosophical Logic, 125–394. Dordrecht: Kluwer.2005.

[Adolphs2011] P. Adolphs, M.Theobald, U.Schafer, H.Uszkoreit, and G. Weikum. Yago-qa: Answering questions by structured knowledge queries. In Proceedings of the 2011 IEEE Fifth International Conference on Semantic Computing, ICSC '11, pages 158–161, Washington, DC, USA. IEEE Computer Society,2011.

[Abujabal2017] A. Abujabal, M. Yahya, M. Riedewald, and G. Weikum. Automated template generation for question answering over knowledge graphs. in Proceedings of the 26th International Conference on World Wide Web, pp. 1191{1200, International World Wide Web Conferences Steering Committee, 2017.

[Zheng2018] W. Zheng,J. Xu Yu, L. Zou, and H.Cheng. Question Answering Over Knowledge Graphs: Question Understanding Via Template Decomposition. *PVLDB*, 11 (11): 1373-1386, 2018.

[Dima2013] C. Dima. Intui2: a prototype system for question answering over linked data. In: Proceedings of the question answering over linked data lab (QALD-3) at CLEF.2013.

[Dima2014] C. Dima. Answering natural language questions with Intui3. In Working Notes for CLEF 2014 Conference, 2014.

[2] We use the following prefixes: dbo for http://dbpedia.org/ontology/, dbp for http://dbpedia.org/property/, and res for http://dbpedia.org/resource/.

[3] https://wiki.dbpedia.org/

[Tripodi2013] R. Tripodi and R. Delmonte. From logical forms to SPARQL query with GETARUNS. New Challenges in Distributed Information Filtering and Retrieval, 2013.

[Shekarpour2013] S. Shekarpour, E. Marx, A-C Ngonga Ngomo, and S. Auer. SINA: Semantic interpretation of user queries for question answering on interlinked data. J. Web Sem. 30 (2015), 39–51,2015.

[Song2015] D. Song, F. Schilder, C. Smiley, C. Brew, T. Zielund, H. Bretz, R. Martin, C. Dale, J. Duprey, T. Miller, and J. Harrison. TR discover: A natural language interface for querying and analyzing interlinked datasets. In ISWC, pages 21{37, 2015.