# Extending DUDES for Ranked Template Generation

Hyunwhan Joe[1], Sungkwon Yang[1], Yongsun Shim[1], Sueun Jang[1], Hong-Gee Kim[1]

[1] Biomedical Knowledge Engineering Laboratory,
Seoul National University, Seoul, Korea
{hyunwhanjoe, sungkwon.yang, yongsun0926,
jchr119, hgkim}@snu.ac.kr

**Abstract.** Question answering systems for Linked Open Data represented in RDF has received attention lately. These systems allow users to access datasets without any prior knowledge of the data model, schema, or query language. Template generation is one such method used by systems to transform natural language questions into SPARQL queries. TBSL is a representative of systems that use template-based approaches. TBSL first transforms questions into an intermediate semantic representation. After this the semantic representation is transformed into SPARQL templates. Several candidate templates can be generated. In this paper we propose an example of a possible scoring method on the intermediate semantic representations that can be later used for ranking the templates.

**Keywords:** Question Answering, Semantic Web, Natural Language Patterns.

## 1 Introduction

There is a large amount of RDF data interlinked together as Linked Open Data (LOD). The problem is that end-users interested in this data have to be familiar with Semantic Web technologies to be able to access them. Question answering (QA) systems are one solution to this problem. QA systems allow users to access datasets without any knowledge of RDF, vocabularies, and SPARQL. One approach to QA is a template-based approach. A *template* is a SPARQL template that represents the general structure of the intended query. The template is not a full SPARQL query and has *slots* which contain information about what kind of entity will fill the slot (resource, class, or property) and the matching lexical term. An example of a template can be seen in Fig. 1. A representative QA system of the template-based approach is TBSL [1].

TBSL consists of three major modules in order; template generation, entity linking, and query filtering and ranking. A natural language question is the input for the template generation module and the output is one or more templates. The templates are the input for entity linking and after the slots are filled with candidate entities and the output of the module is candidate SPARQL queries. The queries are then filtered and ranked where the highest ranked query will be used to retrieve the answer from the dataset.

```
• SELECT ?x WHERE {
        ?x ?prop ?y .
        ?y rdf:type ?cls .
}
  ORDER BY DESC(COUNT (?y)) LIMIT OFFSET 0
  Slots:
• ?cls: CLASS {films}
• ?prop: PROPERTY {produced}
```

**Fig. 1.** An example of a template with slots for the question "Who produced the most films?"

The query filtering and ranking module is needed because the template generation module can produce several templates which leads to several queries. In this paper, we address the query ranking issue by adding possible ranking scores to the candidate templates generated. The intuition is that certain templates tend to be used more often than others depending on the grammar patterns of the question. The paper is an on-going work and presents preliminary results. Section 2 will go more into detail about the template generation process that is needed for the next section. Section 3 will give an example of a possible template ranking score.

## 2 Template Generation

The idea behind TBSL is that the structure of the SPARQL query is decided by the syntax and the *domain-independent expressions* in the natural language question. The SPARQL equivalent of these expressions are the same throughout any dataset which is why they are considered domain-independent. Examples of this are question words such as who, what, where, and when. During the template generation process, the natural language question is parsed into its syntactic structure. TBSL uses Lexicalized Tree Adjoining Grammar (LTAG) [2] for parsing but for ease of explanation, we will be using in this paper dependency and constituency parsing together instead.

A template is not directly generated from the parse tree of the question. It is first transformed into an intermediate representation which captures the semantics of the original question. The intermediate representation is DUDES [3], a variation of Underspecified Discourse Representation Structures (UDRS) [4]. A template is then formed from this DUDES.

Each node on the dependency parse tree of the question has a corresponding DUDES which is the semantic representation of that node. Each DUDES also have additional constituent constraints. The DUDES for domain-independent expressions are defined manually beforehand while the DUDES of domain-dependent expressions are built automatically based on POS tags. Named entities have the DUDES equivalent of resources. Nouns are either represented as class DUDES or property DUDES.

Verbs are represented as property DUDES and empty DUDES which assumes that the property slot comes from a noun elsewhere.

The final DUDES representing the semantics of the question is formed by starting from the bottom node of the dependency tree and merging its DUDES with the DUDES above if the dependency relations match. This will form a new DUDES which will merge with the DUDES above and this will continue till there is no more DUDES to merge. An example of this merging process can be seen in Fig. 2. Since there can be possibly more than one DUDES per node, more than one final DUDES can be made. This also leads to several templates being generated.

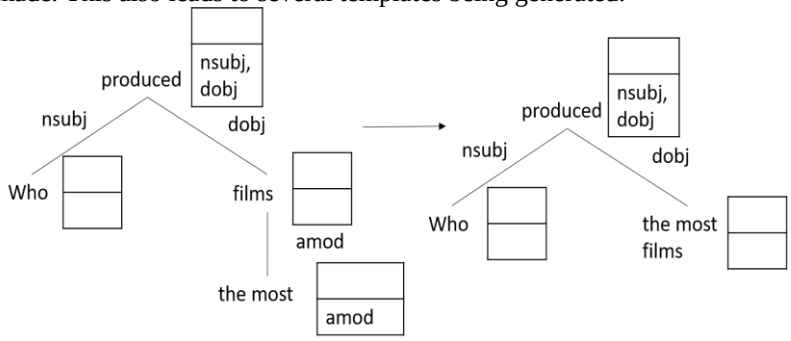


**Fig. 2.** An example of DUDES merging for the question "Who produced the most films?"

## 3 Template Ranking

In this section we continue to use the question "Who produced the most films?" as an example for possible template scoring. This question leads to three possible DUDES. The main reason for this is that the "produced" node has three possible DUDES defined. The first is where the verb "produced" is interpreted as a property. The two other DUDES assume that the property is contributed by a noun elsewhere. An observation from this is that "produced" alone has three possible DUDES but with more context we can see that the first DUDES is more likely to be correct. The context is that when a verb is followed by "the most" and a noun we can assume that the verb is behaving as a property. An exception to this would be if the verb is "has" where it is not behaving as a property but the noun after will. This won't be an issue since "has" is a domain-independent expression which will be defined beforehand and the sentence will not be interpreted as a verb + "the most" + noun sentence.

An example of how context scoring can be used is if each DUDES starts with a default score such as zero. After this each DUDES has grammar conditions that if met they will increase the score of the template such as adding one. The DUDES where "produced" is interpreted as a property would have a grammar condition such as if "the most" is followed by a noun then when it merges with another DUDES the resulting DUDES would have a higher score than the DUDES where "produced" is considered empty. In the example such as in Fig. 3 the DUDES for "films" and "the most" will merge into a "the most films" DUDES. After it will merge with the "pro-

duced" DUDES and the resulting DUDES will have a higher score. The final DUDES score will carry on to the generated template which will be used later during query filtering and ranking.

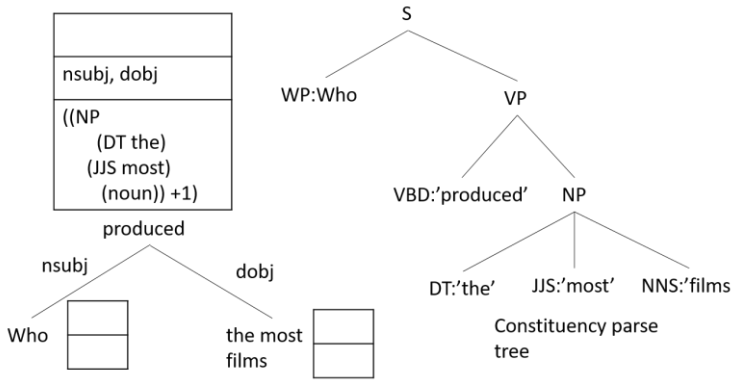


**Fig. 3.** An example of possible template scoring for the question "Who produced the most films?"

## 4 Conclusion

In this paper we proposed a possible scoring method to score templates generated from a template-based QA system. We used the architecture from TBSL which uses DUDES as an intermediate semantic representation for our paper. The final DUDES will be scored based on context. Certain natural language patterns will be given higher scores because they are more likely representations of the question. The generated templates from the DUDESs will carry these scores which will be used later for query filtering and ranking.

**Acknowledgments.** This work was supported by Institute for Information & communications Technology Promotion(IITP) grant funded by the Korea government(MSIP) (No. 2013-0-00109, WiseKB: Big data based self-evolving knowledge base and reasoning platform).

## References

1. Unger, C., Bühmann, L., Lehmann, J., Ngomo, A., Gerber, D., Cimiano, P.: Sparql template-based question answering. In: the 21st international conference on World Wide Web, (2012).
2. Schabes, Y.: Mathematical and Computational Aspects of Lexicalized Grammars. PhD thesis, University of Pennsylvania, (1990).
3. Cimiano, P.: Flexible semantic composition with DUDES. In: The Eighth International Conference on Computational Semantics, (2009).
4. Reyle, U.: Dealing with ambiguities by underspecification: Construction, representation and deduction. Journal of Semantics, 10(2):123-179, (1993).