Demoing Platypus – A Multilingual Question Answering Platform for Wikidata

Thomas Pellissier Tanon[1,2], Marcos Dias de Assunção[1], Eddy Caron[1], and Fabian M. Suchanek[2]

[1] Université de Lyon, ENS de Lyon, Inria, CNRS, Univ. Claude-Bernard Lyon 1, LIP
[2] LTCI, Télécom ParisTech

Abstract. In this paper we present Platypus, a natural language question answering system on Wikidata. Our platform can answer complex queries in several languages, using hybrid grammatical and template based techniques. Our demo allows users either to select sample questions, or formulate their own – in any of the 3 languages that we currently support. A user can also try out our Twitter bot, which replies to any tweet that is sent to its account.

1 Introduction

Recent years have seen the rise of systems that can answer natural language questions such as "Who is the president of the United States?". These systems usually rely on *knowledge bases* (KBs) – large, structured repositories of machine-readable facts. In this paper, we propose to demonstrate a question answering system called Platypus. It differs from existing systems in two main aspects. First, it explicitly targets Wikidata, which is one of the largest general purpose knowledge bases. Second, it supports multiple natural languages with minimal adjustments. Our system is available online as an API[3], as a Web interface[4], and as a Twitter bot[5]. A full description of the system is available as a technical report [1].

2 Related Work

Question Answering (QA) has been an active domain of research since the early 1970's. One group of approaches is based on the grammatical structure of the questions, among which one of them [2] uses the formal grammar of the target language. Our work, in contrast, relies solely on semantic parsing models, and does not need expensive adaptations for different languages. Furthermore, the work of [2] targets only domain-specific knowledge, while we target a knowledge base like Wikidata. Another grammar-based approach employs a multilingual semantic parser on top of universal dependencies [3]. Our work differs from this approach, in that (1) we provide a working implementation on Wikidata and (2) our logical representation matches directly the target KB, so that we can work with specific output types such as strings, dates, numbers, and quantities.

[3] https://qa.askplatyp.us
[4] https://askplatyp.us
[5] https://twitter.com/askplatypus

Our approach differs from [4] in that (1) we provide a method for question answering rather than dispatching to an external service, and (2) we support multiple languages.

Another set of approaches [5–9] relies on machine learning or neural networks in order to build a KB query directly from the question, sometimes using templates. Compared to this previous work, Platypus provides a similar end-to-end learning approach based on templates, but also a grammar-based approach that can answer questions for which there is no template. Furthermore, our work provides easy support for multiple languages, a task that is only starting to be tackled by other works [10, 11].

3 Platypus System

Knowledge Base. Platypus works on a knowledge base. We choose Wikidata for two reasons. First, it provides a large set of lexical representations for its properties, in numerous languages [12] (e.g., "was born in", "has the birthplace", and "est né à" for `bornIn`). Second, Wikidata is one of the largest general purpose knowledge bases on the Semantic Web. We built a specialized service that can perform fast entity-lookups on Wikidata, with support for edit distance and type constraints. In order not to overload the Wikidata SPARQL endpoint, Platypus has its own data storage. To keep our answers accurate, we perform a daily replication of Wikidata to include updates.

Logical Representation. We represent questions not directly in SPARQL, but rather in a custom logical representation. The representation is inspired by dependency-based compositional semantics [13, 14], and adapted to work with multiple languages. The advantage of this approach is that it allows the composition of partial representations. For instance, one could give the representation $\{x \mid \langle \texttt{dynamite}, \texttt{inventor}, x\rangle\}$ for "the inventor of dynamite" and $\{y \mid \langle x, \texttt{birthPlace}, y\rangle\}$ for "Where was X born?"; composing the two thus gives the representation $\{y \mid \exists x\ \langle \texttt{dynamite}, \texttt{inventor}, x\rangle \wedge \langle x, \texttt{birthPlace}, y\rangle\}$ for "Where was the inventor of dynamite born?".

"Where was the inventor of dynamite born?"

$\downarrow$ grammatical analysis

```
1   Where    ADV                             PronType=Int                            0 ROOT
2    was     VERB Mood=Ind—Number=Sing—Person=3—Tense=Past—VerbForm=Fin   1  cop
3    the     DET                  Definite=Def—PronType=Art                  4  det
4 inventor   NOUN                            Number=Sing                             1 nsubj
5    of      ADP                                  -                                  6  case
6 dynamite   NOUN                            Number=Sing                             4 nmod
7   born     NOUN                            Number=Sing                             1 nmod
```

$\downarrow$ semantic analysis

$$\{y \mid \exists x\ \langle \texttt{dynamite}, \texttt{inventor}, x\rangle \wedge \langle x, \texttt{birthPlace}, y\rangle\}$$

$\downarrow$ conversion to database request

```
SELECT ?x WHERE { wd:Q80728 wdt:P61 ?y . ?y wdt:P19 ?x }
```

$\downarrow$ request execution

Stockholm

Fig. 1. Pipeline execution using the grammatical analyzer

Platypus System. Our system takes as input a natural language question, and produces as output a set of answers (RDF terms) from the KB. For this purpose, the question is first transformed into one or several internal logical representations. We provide two different *analyzers* to this end: a primary one (the *grammatical analyzer*) and a secondary one (the *template-based analyzer*). Figure 1 shows the process with the grammatical analyzer.

Grammatical analyzer. The grammatical analyzer takes as input a natural language question, and translates it into a logical representation. For this purpose, it first parses the question with CoreNLP [15], or Spacy [16], yielding a dependency tree. Then it transforms this tree to the logical representation using manually designed rules. For example, we have the following rule:

$$parse\left(\begin{array}{l}\texttt{Where}\\ \quad\big|\,\text{nsubj}\\ \quad\text{X}\end{array}\right) = \big\{p \mid \exists x.\, x \in parse_t(X) \wedge \langle x, \texttt{located in}, p\rangle\big\}$$

This rule allows us to parse the question “Where is Paris?” as follows:

$$parse\left(\begin{array}{ccc} & \text{Where} & \\ \text{cop}\diagup & & \diagdown\text{nsubj} \\ \text{is} & & \text{Paris}\end{array}\right) = \left\{p \,\middle|\, \exists x. \begin{array}{c} x \in parse_t(Paris) \wedge \\ \langle x, \texttt{located in}, p\rangle \end{array}\right\}$$
$$= \{p \mid \langle \texttt{Paris}, \texttt{located in}, p\rangle\}$$

Entity lookup is done using a special rule that returns the set of entities matching a given label. When several rules can be applied, the analyzer returns several results. Hence, *parse* does not return a single logical representation but a set of possible representations. We will discuss below how to filter out the wrong representations. Our rules depend as much as possible on the set of POS tags and the set of dependency tags and not on the input language. Both tag sets are language independent. When specific words are needed, e.g. for connection words (such as “in” or “from”) and question words (such as “where” or “when”), we use dictionaries. We have developed dictionaries for English and French, which allows Platypus to answer questions in these two languages. Support of Spanish is currently in development and German support is planned. Adding support of a new language only requires to fill the dictionaries and make sure that all grammatical constructions of the language are covered by the existing rules.

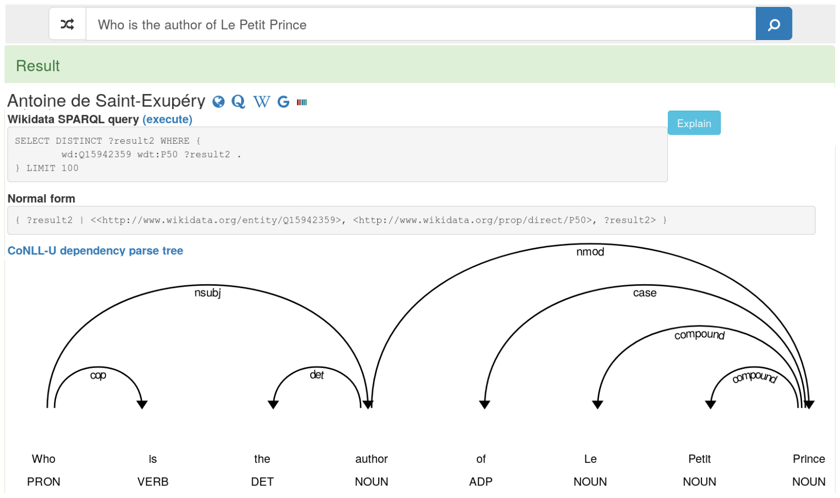
Template analyzer. The second Platypus analyzer is based on templates and implemented using the RasaNLU [17] library. A template is a logical representation with free variables, annotated with natural language questions. For example $\{o \mid \langle\, \text{s}\, , \texttt{birth date}, o\rangle\}$ could be annotated with “When was George^s Washington^s born?”. Our analyzer uses these templates in order to find the logical representation of a given natural language question. For this purpose, the analyzer first finds the template that best matches the question. This is done using a classifier. We encode the question by the average of the word embeddings of its words, and classify them with a linear support vector machine. After this, the analyzer fills the logical representation slots. We use conditional random fields [18] to recognize entity labels in the input sentence, and we match them with the knowledge base entities. We trained this analyzer in English using the WikidataSimpleQuestions dataset [19].

Query execution. We execute the two analyzers in parallel. The grammatical one is executed for all languages and has the advantage of supporting complex sentences. The template-based one works only in English, but has the advantage of working well with short sentences that are not covered by the implemented grammatical rules.

We rank the logical representations according to their likelihood of being the correct interpretation of the question. For this, we take into account the prominence of the mentioned entities in the KB. Finally, the representations are converted into SPARQL, and executed one after the other on Wikidata, until one of them yields an answer.

4 Demonstration setting

Our system is available online at https://askplatyp.us (see Figure. 2) and executes the two analyzers. During the demo session, users can either choose from a set of predefined questions, or ask Platypus any question they want. Since Wikidata is quite exhaustive, it is likely that Platypus can answer questions about the city where the user was born, about the author of their favorite books, or the children of a given president. Users can choose to ask in any of the 3 languages that we currently support. The demo interface shows the grammatical analysis of the question, the logical representation, the SPARQL query, as well as the answer to the question – allowing the user to trace the entire process of question analysis in Platypus.



Who is the author of Le Petit Prince

Result

Antoine de Saint-Exupéry

Wikidata SPARQL query (execute)

```
SELECT DISTINCT ?result2 WHERE {
        wd:Q15942359 wdt:P50 ?result2 .
} LIMIT 100
```

Normal form

```
{ ?result2 | <<http://www.wikidata.org/entity/Q15942359>, <http://www.wikidata.org/prop/direct/P50>, ?result2> }
```

CoNLL-U dependency parse tree

Fig. 2. Platypus Web interface

For those conference attendees who cannot make to our demo, we offer a special service: Platypus can also answer questions via Twitter. For this, the user has to send their natural language question to the Twitter handle `@askplatypus`.

5 Conclusion

We demonstrate a fully functional multilingual natural language question answering system for Wikidata. Our system can work with both a grammatical analyzer and a template-based analyzer to parse natural language questions. During the demo session, Platypus can be tried out in English, French, and Spanish on our Web page https://askplatyp.us – or via our Twitter bot.

Acknowledgments. We thank the contributors of the first version of the Platypus project: M. Chevalier, R. Charrondière, Q. Cormier, T. Cornebize, Y. Hamoudi, V. Lorentz. Work supported by the LABEX MILYON (ANR-10-LABX-0070).

References

1. Pellissier Tanon, T., Dias De Assuncao, M., Caron, E., Suchanek, F.: Platypus – A Multilingual Question Answering Platform for Wikidata. Technical report (2018)
2. Marginean, A.: Question answering over biomedical linked data with grammatical framework. Semantic Web **8**(4) (2017)
3. Reddy, S., Täckström, O., Petrov, S., Steedman, M., Lapata, M.: Universal semantic parsing. In: EMNLP. (2017)
4. Athreya, R.G., Ngomo, A.C.N., Usbeck, R.: Enhancing community interactions with data-driven chatbots – the dbpedia chatbot. In: WWW demo. (2018)
5. Unger, C., Bühmann, L., Lehmann, J., Ngomo, A.N., Gerber, D., Cimiano, P.: Template-based question answering over RDF data. In: WWW. (2012)
6. Sorokin, D., Gurevych, I.: End-to-end representation learning for question answering with weak supervision. QALD-7 (2017)
7. Yahya, M., Berberich, K., Elbassuoni, S., Ramanath, M., Tresp, V., Weikum, G.: Natural language questions for the web of data. In: EMNLP-CoNLL. (2012)
8. Bordes, A., Usunier, N., Chopra, S., Weston, J.: Large-scale simple question answering with memory networks. CoRR (2015)
9. Yin, W., Yu, M., Xiang, B., Zhou, B., Schütze, H.: Simple question answering by attentive convolutional neural network. In: COLING. (2016)
10. Hakimov, S., Jebbara, S., Cimiano, P.: AMUSE: multilingual semantic parsing for question answering over linked data. In: ISWC. (2017)
11. Diefenbach, D., Singh, K., Maret, P.: Wdaqua-core0: a question answering component for the research community. QALD-7 (2017)
12. Kaffee, L., Piscopo, A., Vougiouklis, P., Simperl, E., Carr, L., Pintscher, L.: A glimpse into babel: An analysis of multilinguality in wikidata. In: OpenSym. (2017)
13. Zelle, J.M., Mooney, R.J.: Learning to parse database queries using inductive logic programming. In: AAAI. (1996)
14. Liang, P., Jordan, M.I., Klein, D.: Learning dependency-based compositional semantics. Computational Linguistics **39**(2) (2013)
15. Chen, D., Manning, C.D.: A fast and accurate dependency parser using neural networks. In: EMNLP. (2014)
16. Honnibal, M., Johnson, M.: An improved non-monotonic transition system for dependency parsing. In: EMNLP. (2015)
17. Bocklisch, T., Faulker, J., Pawlowski, N., Nichol, A.: Rasa: Open source language understanding and dialogue management. CoRR (2017)
18. Lafferty, J.D., McCallum, A., Pereira, F.C.N.: Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In: ICML. (2001)
19. Diefenbach, D., Pellissier Tanon, T., Singh, K.D., Maret, P.: Question answering benchmarks for wikidata. In: ISWC poster. (2017)