Answering Natural Language Questions on RDF Knowledge base in French

Nikolay Radoev[1], Mathieu Tremblay[1], Michel Gagnon[1], and Amal Zouaq[2]

[1] Département de génie informatique et génie logiciel, Polytechnique Montréal
{mathieu-4.tremblay,nikolay.radoev,michel.gagnon}@polymtl.ca
[2] School of Electrical Engineering and Computer Science, University of Ottawa
azouaq@uottawa.ca

Abstract. While SPARQL is a powerful way of accessing linked data, using natural language is more intuitive for most users. A few question answering systems already exist for English, but none for French. Our system allows a user to query the DBpedia knowledge by asking questions in French, which are automatically translated into SPARQL queries. The system classifies questions by various types that are resolved by type specific heuristics. To our knowledge, this is the first French-based question answering system in the QALD competition.

1 Introduction

As more and more structured data are made available on the Web, giving Web users access to this data is a crucial aspect for the uptake of the Semantic Web. Among the datasets that are openly accessible to the public, we find general knowledge bases (KBs), such as DBpedia [1], which contains information extracted from Wikipedia, and many specialized KBs that have curated domain-specific knowledge, such as Dailymed. However, given their reliance on SPARQL[2], they are difficult to use for the average user. Developing a powerful yet intuitive interface to allow natural language queries is a problem that has brought growing amounts of research in the past years, including in the QALD[3] challenges.

Currently, most existing systems have been focused on English, given its wide popularity and the vast amount of resources available in existing KBs. However, there is a need to handle multilingual queries on the Semantic Web. Thus, in this paper, we have chosen to tackle the French language. French displays some different syntactic structures that do not easily allow the reuse of techniques developed for English. For example, adjective placement in English precedes the noun while French has the adjective, most of the time, following the noun it describes. Verb conjugation is also more complex in French than in English.

Interpreting a question given in a natural language is a well-known but unsolved problem [4]. In general, it requires the extraction of a semantic representation that is the result of a multiple-phase approach. The question must be processed to extract keywords and resources identifiers. Then, those resource identifiers must be mapped to resources in the KBs. A complex task, given the fact that (i) those resources might not exist in the KBs and (ii) natural language syntax creates ambiguities that cannot be resolved without proper context.

Some previous works on the problem used *controlled natural language* (CNL)[5] approaches to restrict grammar and syntax rules of the input question. Such approaches have the merit of reducing ambiguity and increasing the accuracy of the proposed answers. However, we consider that the rigidity of an imposed grammar might seem awkward to an average user and we have opted not to impose any constraints on the questions given as input.

2 System Overview

Our system enables users to input queries in French that are then analyzed and answered with information found in DBpedia. Its main focus is to interpret and answer *Simple Questions*. A *Simple Question* in our system is defined as a question that concerns only one single entity and a single property of this entity. While we are actively working on handling more complex questions involving multiple entity/property relations, we have concentrated our work on the *Simple Questions*. Our approach to the problem is a system that uses syntactic features to identify what information the user is looking for.

The first step of our system is to determine the type and possible sub-type of the question. For now our system supports the following types : *Boolean, Date, Number and Resource*. Additionally, *List and Aggregation* subtypes are added to some of those main types. *List* questions returns answers containing multiple elements and both *Resource and Date* questions can have *List* as a subtype. *Aggregation* questions include (i) questions that require an ascending or descending order and (ii) questions that require aggregation. *Date, Number and Resource* questions are the ones able to have *Aggregation* as a sub-type.

By analyzing the questions given in the 2016 and 2017 train datasets, we have extracted various keywords and patterns that occur most often in a given question type. The extracted patterns rely both on lexical matching and positioning (start of sentence or just their general presence in the question). For example, pronoun inversions such as *existe-t-il(elle)...* (where the general form is *VERB-t-il(elle)*) appear only in close-ended(*Boolean*) questions.

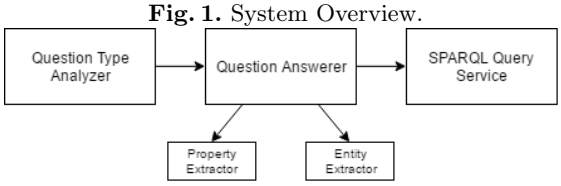
Classification is made by matching the question string against the list of extracted patterns and if a match is found, the question gets assigned a specific type. If no match is found after going through all extracted keywords and patterns, we give it a default value of *Resource* question type. The same method is then applied for the subtypes with a few additional tweaks. For *List* questions, we look for plural question words or verbs and for *Aggregation*, we try to determine whether the answers requires to be sorted in a descending or an ascending order. Again, this classification is made by using a list of keywords and patterns extracted from the datasets.

Once the system knows the question type, the query is sent to specific question type solvers. For instance, a question such as *Is Michelle the name of Barack Obama's wife?* will be sent to the *Boolean* answerer, and *When was Barack Obama born?*, is handled by the date question solver.

Every question solver makes use of one or more submodules that function as *extractors*. There are two main extractors : an *entity extractor* and a *property*

extractor, as seen on Figure 1.1, which are used to identify the entities and properties in a given question. Specific question solvers require specific property extractor heuristics. For example, date-related questions require additional work to extract the correct properties (more details are given in Section 3).

Fig. 1. System Overview.



3 Simple Question Analysis

As previously mentioned, in our current implementation, we deal only with simple questions, limited to at most one entity and one property, such as *Qui est le père de Barack Obama?* (Who is the father of Barack Obama?), where 'Barack Obama' is the entity and 'father of' is the property. In the case of *Boolean* questions, we also treat *Entity - Entity* relations. To extract the entity from the sentence, we use a syntactic parser to identify the noun groups and we then find out the ones that correspond to an existing entity in DBpedia (both French and English version) by looking up the *dbpedia.org/resource/[noun]* URL and retaining only the existing links. Since the question is in French, the *sameAs* link is used to find the corresponding URI in the English version of DBpedia. In order to get better results, we add variations of the identified nouns by applying modifications such as plurality indicators and capitalization. Every modification added reduces the chance of the entity to be chosen as main one. For example, the entity *queen* can also be manipulated in order to extract the entity *Queens*. However, since *Queens* requires 2 modifications (pluralization and capitalization), it is less likely to be selected.

Once the entity is extracted from the sentence, the property is found by removing the entity from the query and analyzing the remaining tokens to find nouns or verbs. We then proceed by trying to find the property in the entity's description in DBpedia using a lexical match. When no link between the extracted entity and property is found, we attempt disambiguation on the entity by following the Wikipage disambiguates link, if such a link is available. These links provide a way to find other entities that can be a possible match. For example, in *Who is the producer of Titanic?*, the entity *Titanic* is found. Using the disambiguates link in *Titanic*, we can find *Titanic(film)*. By removing *Titanic* from the query, *producer* can be discovered through the *Titanic(film)* page.

To facilitate the identification of the predicates used in the SPARQL query that corresponds to the question sentence, we built a lexicon by mapping a

list of common DBpedia properties to French expressions, in addition to manually adding bindings that were not present in the French DBpedia. For example, "http://dbpedia.org/ontology/spouse" and "http://fr.dbpedia.org/property/conjoint" are both mapped to *conjoint* (spouse), *épouse* (wife), *femme* (wife) and *mari* (husband). With such bindings, the system is able to take into account the various ways of expressing the property in French: *Qui est la conjointe/l'épouse/la femme de Barack Obama?* (Who is the spouse/wife/wife of Barack Obama?)

Date Questions
Some queries require that our system find dates for events such as the birth of a person. For most of these questions, the label for the property is not written plainly in the question. For example, *Quand Rachel Stevens est-elle née?* (When was Rachel Stevens born?) is a type of question where the property (birthYear) is not directly given but must inferred. Our solution for this problem is to use pattern matching with words that are often used, such as *"année de naissance"* (year of birth) or *"est né(e)"* (is born). This is then translated to the URI related to the *dateOfBirth* property through our lexicon. Other types of dates, such as death date and end of career date, can also easily be handled in the same way. When the property cannot be inferred, we try using a default *http://dbpedia.org/ontology/date* property.

Boolean Questions
The first step in the processing of *Boolean* questions is to determine whether the question involves other entities, as in *Is Michelle Obama the wife of Barack Obama?* (note that in this example *Barack Obama* is identified as the concerned entity). If it is the case, the program verifies whether a relation exists between the entities and if it is the right type (in this case, a relation of type *spouse*). When the question does not involve other entities, it is about whether a property exists for a specific entity. We can consider the example *Existe-t-il un jeu vidéo appelé Battle Chess?* (Is there a video games called Battle Chess?). In this case, we simply need to find all entities with a label *Battle Chess* and find out if one of them is a video game. Here again we rely on the presence of *Existe* and look for an *rdf:type* relation using a mapping available in our lexicon. The last type of question supported by our system is whether an entity is of a specific type. For instance, *Are tree frogs a type of Amphibian?*. In this case, we extract the entity from the sentence (tree frog), and verify if it has the right type (Amphibian).

Aggregation Questions
Aggregation questions are divided in two different categories. The first category contains questions that are looking for a numeric answer, such as *How many languages are spoken in Colombia?*. In this case, all possible entities and properties are extracted from the question, with entities/properties immediately after the "How many" expression being considered as more likely to be the main subject of the SPARQL query. The second category contains questions that involve ordering result sets in a specific order. Expressions such as *le plus grand* (the

most), *le plus gros* (the biggest), etc., are used to decide the sorting order. After ordering the results, we take into account the offset when the question asks for the *nth* value instead of the first/last one.

List Questions
List questions are those that can, but do not necessarily require an answer containing multiple entries. They can be seen as a subset of aggregation questions, but without additional sorting or counting over the result set. For example, *Who are the founders of DBpedia?* is a list question that returns multiple resources. Given that the type of results can vary, list questions are first analyzed based on their type. We specifically detect particular types, such as list of *Resource or Dates* based on our previous question solvers. If more than one possible results are found following the SPARQL query, all results are added to the final set.

4 Evaluation

To evaluate the efficiency of our system, we ran it on a set of 165 simple questions out of the 211 questions provided as part of the 2017 training sets for QALD. Our system was able to answer 150 questions out of the 165 simple ones and a total of 157 questions. This gives us 90% of accuracy on the simple questions and 74.41% overall. To increase our accuracy, we plan on improving our entity extractors and the way of selecting the right entity. To do so, we plan to translate more labels for entities from French to English, which would allow us to answer questions where we cannot easily find the entity in the French DBpedia. We also plan to process more complex queries involving more than one entity and property.

5 Conclusion

While these results remain modest, and further development is needed for more complex questions, to our knowledge there has never been any system focused on French in the QALD competition. This work is a first step to encourage the development of question answering systems for the French language.

References

1. `http://wiki.dbpedia.org`.
2. `https://www.w3.org/TR/sparql11-overview/`.
3. Vanessa Lopez, Christina Unger, Philipp Cimiano, and Enrico Motta. Evaluating question answering over linked data. *Web Semantics Science Services And Agents On The World Wide Web*, 21:3–13, 2013.
4. Poonam Gupta. A survey of text question answering techniques. International Journal of Computer Applications, 2012.
5. Giuseppe Mazzeio. Answering controlled natural language questions on RDF knowledge bases. `https://openproceedings.org/2016/conf/edbt/paper-259.pdf`, 2016.