LAMA: a Language Adaptive Method for Question Answering

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

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

Abstract. The LAMA system focuses on interpreting natural language questions within a reasonable time. Originally designed to process queries written in French, the system has been redesigned to also function on the English language. A method for decomposing complex queries into a series of simpler queries makes our system better at answering those questions. The use of preprocessed data and parallelization methods helps improve individual answer times.

1 Introduction

The Scalable Question Answering over Linked Data Challenge requires finding a way to transform a question into a query over a knowledge base. Questions range from simple ones, which target a single fact about a single entity, to complex questions that involve many entities and require some kind of computation or inference to provide an answer. The training data provided in the challenge consists of 5000 natural language questions, for which one or more answers must be extracted from the DBpedia knowledge base.

We present LAMA (Language Adaptive Method for question Answering), an extension to AMAL[1], a system that was originally created for the QALD2017[2] Challenge at the ESWC2017[3] conference. The first version of our system was built specifically to handle questions in French. In LAMA, we extend this capability to the English language and handle more complex questions.

Some previous works aimed at Question-Answering systems used *controlled natural language* (CNL)[4] approaches to restrict the syntactic form of the input question. In previous QALD challenges[5], many participating systems defined features or ad-hoc heuristic approaches to translate natural language questions into, mainly, SPARQL queries. The system we propose aims at minimizing the amount of manual features required to process questions. We also aim at optimizing the preprocessing time for each question to improve scalability for a large amount of questions or to deal with complex questions that may potentially be costly in terms of computational resources. Finally, we propose a method to decompose complex questions into a series of smaller, simpler queries that can be then more easily processed.

2 System Overview

Our previous system enables users to input queries that are analyzed and answered based on information found in DBpedia. Originally, its main focus was to interpret and answer *Simple Questions*, which are questions that concern only one single entity and a property of this entity (e.g. *Who is the mayor of Montreal?*) or a single relationship between two entities (e.g *is Valerie Plante the mayor of Montreal?*). The LAMA system also handles some *Complex Questions* by converting them into a set of *Simple Questions* and processing those smaller parts. The architecture of our system involves a straightforward pipeline : *Question Type Classifier*, a *Question Solver* and a *SPARQL Query Generator*.

The first step is to determine the type of the question and the nature of the expected answer. Currently, questions are grouped into the following types: *Boolean, Date, Number* and *Resource*. *Resource* is the most generic type and designates questions for which the expected answer is one or more URIs. It is used as the default type for questions that don't fit any of the other categories. The question type is determined through pattern matching with manually extracted patterns (roughly 10 patterns per question type) from the QALD6 and QALD7 training sets, totaling more than 400 questions (see Table 1). This method has the advantage of being easily transferable between languages in the context of a multilingual system as patterns can be easily expressed in different languages. In fact, after extracting patterns for both French and English, the system was able to accurately predict the question type in more than 92% of the test cases (QALD7 training set).

Table 1. Pattern examples for different question types

Question Type	**English**	**French**
Boolean	Does (start of question)	Est-ce que
Date	When (start of question)	Quand
Number	How many	Combien
Resource	Who is/are/were	Qui est/sont/étaient

Once the system knows the question type, the query is sent to a specific solver that has been designed for this type of question. For instance, a question such as *Is GIMP written in GTK+?* will be sent to the *Boolean* answerer, and *Who died due to Morphine?* is handled by the *Resource* 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*.

Finally, we generate one or more SPARQL queries based on the entities and properties found by the extractors of the solver. Given that multiple valid entities and/or properties can be found in a single query, all possible SPARQL queries based on these valid combinations are generated and saved in a queue. Currently, the system supports *ASK* and *SELECT* queries along with the *COUNT*, *DISTINCT* and *ORDER BY* modifiers.

3 Question Analysis

As previously mentioned, the original AMAL project focused on *Simple Questions*, limited to at most one entity and one property. For example, a simple question is *Who is the father of Barack Obama?*, where 'Barack Obama' is the entity and 'father of' is the property. To extract the entity from the question, we first try to match the words (or a combination of them) to existing DBpedia entities and if none are found, we use the DBpedia Spotlight tool to identify potential ones. In order to increase the set of considered entities, we add variations of the identified nouns using normalization and stemming. Every additional modification reduces the likelihood of the entity to be chosen. For example, the entity *queen* can be transformed to extract the entities *Queens* or *Queen*. However, since *Queens* requires 2 modifications (pluralization and capitalization), it is less likely to be selected as the correct entity by the entity extractor.

Once the entities are extracted from the sentence, the properties are found by removing the entities from the query and analyzing the remaining tokens to find nouns or verbs for potential properties. To increase time performance, we have automatically extracted more than 2600 DBpedia object properties along with their labels, thus building a property lexicon. This allows us to easily find existing properties based on their label. If the extracted entities do not have any of the potential properties in their description, we attempt disambiguation on each entity by following the Wikipage disambiguation link, if such a link is available. For example *Film* leads to the page of *Film* (as in a movie), but disambiguation can lead us to *Plastic Film* for queries such as : *Is Nylon a type of film?*. The original AMAL system relied on manually annotated mappings between properties and entities to tackle ambiguities in property selection. This initial reliance on manual bindings has been reduced in LAMA by adding some basic rules for valid entity-property relations. For example, *the tallest* can be ambiguous given that it can both mean *height* or *elevation*, with the first meaning used in a relation with the DBpedia entity type *Person*, and the second used with entities that are instances of *Natural Place*. A valid rule for *the tallest* would thus be: *{height: Person}, {elevation: NaturalPlace}*. Having such rules can improve performance and speed by exploring only properties that are part of a valid rule.

Complex Question Simplification
Some natural language queries can be more complex, containing multiple relationships between entities in the same sentence. However, many of those *complex* questions can be represented as a chain of smaller, simpler queries. For example, the question : *Who were involved in the wars where Jonathan Haskell battled?* can be transformed into *What battles was Jonathan Haskell in?* and *Who were the combatants in those wars?*. The question can be separated in two sub-queries with *wars* being the only shared word. For now, deciding when to split a *complex question* into multiple *simple questions* is based on the presence of a keyword (where, which, also, whose, etc.), *where* being the separation marker in this example. Processing smaller questions after the split improves accuracy by re-

ducing possible noise when looking for entities or properties in the query. Using the given example, splitting the query enables us to not falsely detect *Jonathan Haskell* and *involved* as the linked entity and property and erroneously use them together to look for an answer.

Simple questions are represented using a single SPARQL triple pattern. When a complex question is broken down into multiple simple ones, the generated RDF triples can be combined into a single SPARQL request to save processing time.

4 Parallel Processing and Scalability

The original version of the AMAL system had a procedural processing cycle which could bring scalability issues. The system had to wait for every SPARQL query sent to DBpedia to finish its execution before trying a new one and there was no predefined timeout for a request. Originally, all answers were expected to be in English, even if the system was designed to handle French queries only, thus requiring an extra layer of translation for every question. Questions where multiple translations were required were the biggest problem, since the Google API does not enable batch queries.

For the SQA challenge, we propose a redesign of the system, focusing on optimizing scalability. The following points are the main focus of the optimization techniques and target different parts of the process in order to maximize results.

- The extra step of translating French to English is no longer required, given that the questions are in English only. This saves around 15 to 20% of computation time, depending on the complexity of the question and the amount of translations that would be necessary to process the equivalent question in French.
- Some parts of the semantic analysis have been moved to the client side, most notably the lookup of valid DBpedia properties, which no longer requires sending requests to DBpedia. This helps with reducing the time spent waiting for a response and since the data is kept in a Hashtable, access complexity is on average O(1).
- SPARQL queries for the same question are ran in parallel. Since the generated SPARQL queries are independent, running a query as soon as it is generated by the solver, without waiting to see if the previous one was successful or not, helps with the scalability of the system for multiple queries ran at the same time.

5 Evaluation

The evaluation of our system was done on the training set for the SQA2018 challenge. The base benchmark was ran on the 5000 provided questions in the training set. The training set contains roughly 32% of *Simple Questions*. We have evaluated both the accuracy of the returned answers and the time to execute all queries. The evaluation was performed on a 4 core i5-4690K CPU using a hard 5 second timeout on SPARQL queries sent to DBpedia's endpoint to avoid

system lockup. The system was able to answer 2654 out of the 5000 questions, which gives us a 53.3% accuracy for the answers with a success rate of 74% for *Simple Questions* and 42.5% for *Complex Questions*. The evaluation took a total of 15603 seconds, giving us an average of 3.12 seconds per query.

While the results are not as good as the original AMAL system (74% accuracy), we have to note that the new version relies much less on case by case exception handling and it has transitioned from a system aimed specifically at the French language to a multilingual system that handles French and English questions. One of the current limitations of the system is its difficulty to correctly split complex questions into multiple simple questions. Most of the errors in complex questions are due to the fact that the keyword used to split the sentence is not in fact a correct separator. For example, the sentence *Name the TV show whose company is HBO and Playtone?* has been incorrectly split into *Name the TV show* and *company is HBO and Playtone?*. We plan on improving those results with the processing of *Complex Questions* using a syntactic analysis and the integration of disambiguation techniques for entities instead of simple URI lookups. We will also work on the automatic enrichment of the currently used multilingual lexicon.

Finally, the SQA challenge also involves evaluating the competing systems on a test set. We observed that, in contrast to the training set, the test set includes many errors such as spelling errors, case errors or incorrectly tokenized questions. We expect a decrease of LAMA's performance due to these errors.

6 Conclusion

The LAMA system is a modification of a currently existing QA system redesigned for multilingual support. We have modified it by adding emphasis on scalability and reduced its dependence on ad-hoc heuristics and case by case exception handling. There is still some work to be done, most notably by improving the way complex queries are split. We also plan to test the system under a much bigger load in order to truly verify its scalability.

References

1. Nikolay Radoev, Mathieu Tremblay, Michel Gagnon, and Amal Zouaq. Amal: Answering french natural language questions using dbpedia. In Mauro Dragoni, Monika Solanki, and Eva Blomqvist, editors, *Semantic Web Challenges*, pages 90–105, Cham, 2017. Springer International Publishing.
2. Qald2017 challenge – eswc 2017 – hobbit. `https://project-hobbit.eu/challenges/qald2017/`. (Accessed on 03/29/2018).
3. 14th eswc 2017 |. `https://2017.eswc-conferences.org/`. (Accessed on 03/29/2018).
4. Giuseppe Mazzeio. Answering controlled natural language questions on RDF knowledge bases. `https://openproceedings.org/2016/conf/edbt/paper-259.pdf`, 2016.
5. Dennis Diefenbach, Andreas Both, Kamal Deep Singh, and Pierre Maret. Towards a question answering system over the semantic web. *CoRR*, abs/1803.00832, 2018.