Closed Domain Question Answering for Cultural Heritage

Bernardo Cuteri

DEMACS, University of Calabria, Italy
cuteri@mat.unical.it

Abstract. In this paper I present my research goals and what I have obtained so far into my first year of PhD. In particular this paper is about a novel architecture for closed domain question answering and a possible application in the cultural heritage context. Unlike open domain question answering, which makes intensive use of Information Retrieval (IR) techniques, closed domain question answering systems might be built on top of a formal model with the possibility to apply formal logics and reasoning. Natural language question answering poses some non-trivial problems to tackle. We investigate such problems and propose some solutions based on AI techniques, picking the Cultural Heritage domain as a target application.

Keywords: Closed domain question answering, AI, NLP, ASP, cultural heritage

1 Introduction

The information need of a user often resolves in a simple question where it would be useful to have brief answers instead of whole documents to look into. IR techniques have proven to be very successful at locating relevant documents to the user query into large collections, but the effort of looking for a specific desired information into such documents is then left to the user. Question answering attempts to find direct answers to user questions.
As the intuition says, answering to any kind of question, with no linguistic and no domain restriction is a very hard task. When no restriction is made on the domain of the questions we are talking about open domain question answering. Instead, when questions are bound to a specific domain we are talking about closed (or restricted) domain question answering (CDQA).
In open domain QA, most systems are based on a combination of Information Retrieval and NLP techniques[3]. Such techniques are applied to a large corpora of documents: first attempting to retrieve the best documents to look into for the answer, then selecting the paragraphs which are more likely to bear the desired answer and finally processing the extracted paragraphs by means of NLP. Such approach is also behind many closed domain question answering systems, but in this context we might benefit of existing structured knowledge.
Some of the very early question answering systems were designed for closed

domains and they were essentially conceived as natural language interfaces to databases [1][2].
The idea of studying and applying closed domain question answering for the cultural heritage domain comes from the PIUCULTURA project, which is a project of which my university is a research partner. This project aims at implementing a mobile system for cultural heritage fruition. My university is in charge of research and develop techniques for the implementation of a question answering prototype for cultural heritage. For what concerns closed domains, cultural heritage can benefit of structured data sources: in this context, information has already started to be saved and shared with common standards. One of the most successful standard is the CIDOC Conceptual Reference Model. The CIDOC-crm provides a common semantic framework for the mapping of cultural heritage information and can be adopted by museums, libraries and archives.
Our idea is to design and implement a system capable of interpreting natural language questions regarding cultural heritage objects and facts, map the input questions into formal queries compliant to the CIDOC-crm model and execute such queries to retrieve the desired information.
In closed domains, question structures are more predictable than in open domain and we propose to design a sophisticated module of template matching based on a declarative formalism (Answer Set Programming [5]) for question classification and query extraction. A particular feature we want to introduce is the possibility to have dialogues instead of only atomic questions. This might help when the initial question is ambiguous or the system needs more clarifications to provide an accurate answer. Also, the fact that the system is based on formal queries rather than statistical methods might lead to a more robust answer creation, with the possibility to obtain a step-by-step justification of the answer and an easier validation. In the following sections we present an architectural model of the system and provide some more details about the tasks involved in the question answering process.

2 System Architecture and Working Principles

Figure 1 shows the architecture of the system (with some simplifications) highlighting the main modules and how the process looks like. The process is split in five tasks:

1. question processing
2. template matching
3. query expansion and contextualization
4. query execution
5. answer creation

In the following subsections we are going to break through the process and analyze it step by step.

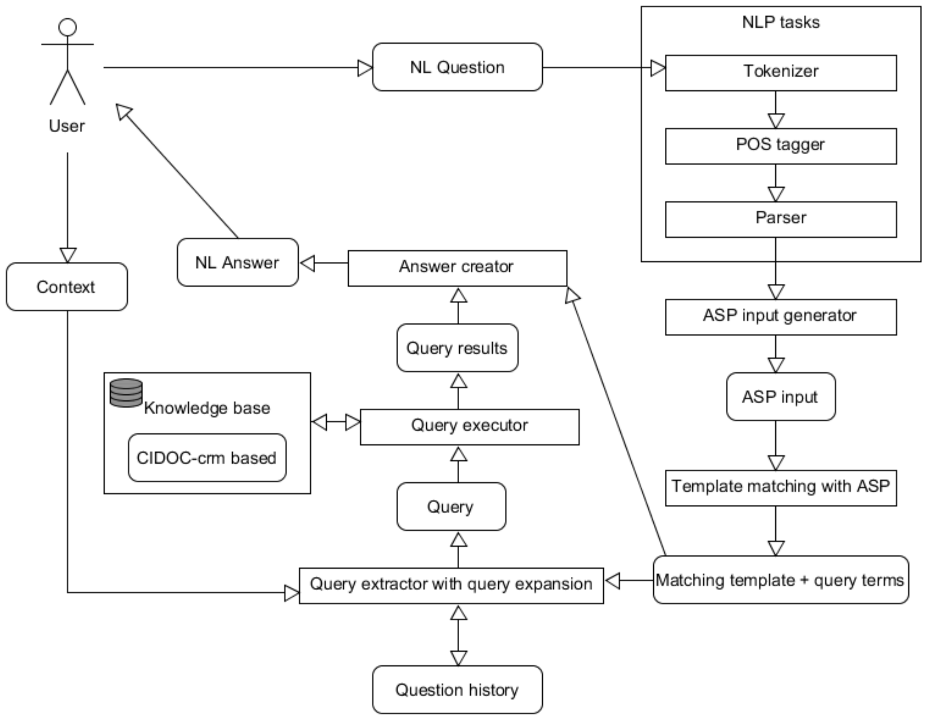

Fig. 1. Simplified architecture with single question interaction

2.1 Question Processing

This is the main NLP step. Fortunately, tokenization, POS tagging and natural language parsing have decades of research behind and there are plenty of tools around that are able to solve such problems efficiently. With respect to the Cultural Heritage domain, something we can not overlook is the importance of entity recognition as we might have proper nouns of artefacts or persons that must not be mistakenly treated by NLP tools (e.g. splitting the title of a painting into distinct grammatical parts).
First the question is tokenized and tagged with part-of-speech (POS) tags. Then a natural language parser is in charge of extracting grammatical relations (a.k.a. typed dependencies) from the text (e.g. who is the subject of what verb, what is the object and so on).

2.2 Template Matching

Questions are classified and transformed into formal queries by means of template matching. In this context, templates represent the structure of typical questions. If a certain template is matched we can infer something about the question type. Every question template is accompanied with a formal query in which some *slots* are empty and are filled with terms extracted from the question that matches the template. For example, imagine we have the template for questions of the type *Who verb object*: the question *Who painted Guernica?* matches such pattern and a corresponding query can be created.*Guernica* and *painted* might then be used as constants in the query, filling the empty *slots* mentioned before.
We want to investigate the possibility to implement template matching with Answer Set Programming [5](ASP). ASP evolved from deductive databases, logic programming and nonmonotonic reasoning. It is a flexible language for knowledge representation and reasoning, and for declarative problem solving, and efficient systems are available[6]. ASP is thus a concrete tool for developing complex applications by just specifying a set of logic rules of the form $Head :- Body$, where *Body* is a conjunction of possibly negated atoms, and $Head$ is a disjunction of atoms. The stable models, or answer sets, of an ASP program correspond to the solutions of the modelled problem. The programmer does not need to provide an algorithm for solving a problem with ASP; rather, she specifies the properties of the desired solution for its computation by means of a collection of logic rules called logic program. The stable models or answer sets of an ASP program correspond to the solutions of the modelled problem. The logic rule is an expression that looks like $Head :- Body$, where *Body* is a logic conjunction possibly involving negation, and $Head$ is either an atomic formula or a logic disjunction. The language of ASP, besides disjunction in rule heads and nonmonotonic negation in rule bodies, features also special atoms for defining aggregates, strong constraints for selecting solutions, and weak constraints for solving optimization problems. The implementation details go beyond the scope of this paper, but we can say that ASP is a good candidate for a fast and declarative

implementation of template matching. This step requires a small preprocessing step in which the input (i.e. words and associated grammatical relations and parts-of-speech) is transformed into ASP facts. A simple example of a possible template for matching questions of the type *Who-verb-object* is the following:

```
template(1, bt(W1,W2)):- textWord(1, who), gr(2,3,dobj),
    textWord(2,W1), textWord(3,W2).
```

Where the *textWord* predicate denotes the presence of a certain word in a certain position and the *gr* predicate denotes a grammatical relation between two words. *gr(2,3,dobj)* means that word in position 3 is the object of word in position 2. This template is a bit simplified and does not take POS tags into account, but gives an idea of how to implement a template in ASP.

2.3 Query Expansion and Contextualization

The template matching result is a formal query. Sometimes, to be effective, the query has to be expanded with context information and/or word semantic information. We can try to understand the importance of those two by providing an example.
Let's say that we asked *When was the monalisa created?*. An admissible following question could be *And who did it?*. The pronoun *it* clearly stands for the painting, but in order to understand it, the system has to get context information or at least store question histories. Another problem that we can analyze with the previous example is the following: let's say that our knowledge base contains the information *Leonardo da Vinci painted the monalisa*. We know that, if someone *painted* something we can also say that they *did* it. The question answering system has to deal with such and other similar problems. A possible solution is to expand the query by using synonyms, hyperonyms and other word semantic relations. Fortunately, there are some available encyclopedic dictionaries that are able to provide such relations. Among them there is BabelNet[4] which has also the desirable property of being multilingual and this might help in case we want to extend the work to different languages.

2.4 Query Execution and Answer Creation

In our model, the query is executed against a structured knowledge base. Query results (if any) can then be used to build a natural language answer with a mechanism similar to template matching, but in the inverse direction. A possible approach is that each question template is paired to an answer template. The answer template may have empty slots for answer terms and it is used by the answer creation module to build the NL answer once the query has been executed successfully.

3 Current State and Future Works

In this paper we presented an architecture and some implementation ideas for a closed domain question answering system and discussed about the tasks involved in the process. The work is currently under development, studies have been conducted to investigate current research trends in question answering and available solutions. At this moment we have developed a small QA prototype capable of answering simple questions. It uses the Stanford parser[7] for tokenization, POS-tagging and parsing, integrates Babelnet[4] for query expansion, and supports some common question types. For example it is possible to ask *who* performed a certain *action* on a certain *object*, or where a certain *object* is located. Templates cover different ways to express the same question like *where is Guernica located?* or *in what museum is Guernica located?*. We are investigating on how to cope with question nuances, trying to design some more general templates for questions that are not perfectly matched by a simpler template. This is where ASP (with disjunction, weak constraints and aggregates) might play a crucial role as opposed to less expressive languages like Datalog.
We started to create a broad catalogue of possible questions in order to create more complex templates and extend the system to adapt to more difficult questions. We are now planning to extend this approach to its limits, trying to manage a broad set of questions on cultural heritage. If it works fine we also plan to add multilingual support checking if the template system is easy to extend to different languages. We also want to investigate how to implement non-trivial dialogues centered around questions instead of only single atomic questions.

References

1. Woods, W. A. (1973, June). Progress in natural language understanding: an application to lunar geology. In Proceedings of the June 4-8, 1973, national computer conference and exposition (pp. 441-450). ACM.
2. Green Jr, B. F., Wolf, A. K., Chomsky, C., Laughery, K. (1961, May). Baseball: an automatic question-answerer. In Papers presented at the May 9-11, 1961, western joint IRE-AIEE-ACM computer conference (pp. 219-224). ACM.
3. Hirschman, L., Gaizauskas, R. (2001). Natural language question answering: the view from here. natural language engineering, 7(04), 275-300.
4. Navigli, R., Ponzetto, S. P. (2012). BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. Artificial Intelligence, 193, 217-250.
5. Gelfond, M., Lifschitz, V. (1991). Classical negation in logic programs and disjunctive databases. New generation computing, 9(3-4), 365-385.
6. Calimeri, F., Ianni, G., Ricca, F., Alviano, M., Bria, A., Catalano, G., ... Manna, M. (2011, May). The third answer set programming competition: Preliminary report of the system competition track. In International Conference on Logic Programming and Nonmonotonic Reasoning (pp. 388-403). Springer Berlin Heidelberg.
7. Klein, D., Manning, C. D. (2003, July). Accurate unlexicalized parsing. In Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1 (pp. 423-430). Association for Computational Linguistics.