Coupling Natural Language Interfaces to Database and Named Entity Recognition

Ana Guimarães

L^{2}F - Spoken Language Laboratory, INESC-ID Lisboa

Abstract. A Natural Language Interface to a database has to be able to precisely identify the entities that it is asked about. In the particular situation of a database such as IMDB, more than 2.500.000 movies, actors and staff names may appear in a question. In this context, Named Entities Recognition can be a heavy task. In this paper, we show how to use the database itself to find out the Named Entities and we analyse the pros and cons of our approach.

1 Introduction

Question/Answering (QA) systems are a long-standing goal for Artificial Intelligence (AI) research and the appearance of the first Natural Language Interfaces to Databases (NLIDB) in the 70's represent their first steps. Nowadays, with the growth of the available digital information, as well as the increasing amount of people accessing information through their web browsers, PDAs, and cell phones, the need for QA systems has become even more acute.

Following this needs, if in the early stages of AI, QA systems were using the structured knowledge from the databases to find the answer to a question, currently, the concept of QA gained a broader sense as they no longer find the answers (only) in databases, but (also) within large collections of open-domain documents.

Supporting this trend, within the last years, evaluation forums with QA tasks, such as NTCIR[1], TREC[2] and CLEF[3], that pretend to improve the research in this field, by evaluating the competing systems in the same conditions, have been growing in participation.

Nevertheless, despite the importance of non restricted QA systems, specific domains QA applications have also gained popularity. In fact, as said in [1], dealing with the incorporation of domain-specific information into QA techology, and with the possibility of reaching deep reasoning capabilities, Restricted QA systems can make the bridge between structured knowladge-based and free text-based QA. In this paper, we will present a NLIDB, that is part of a general QA system.

[1] NII Test Collection for IR Systems (`http://research.nii.ac.jp/ntcir/`)
[2] Text REtrieval Conference (`http://trec.nist.gov/`)
[3] Cross-Language Evaluation Forum (`http://www.clef-campaign.org/`)

In the late 90s, the authors developed a NLIDB that was answering questions about tourist resources [2]. It was a traditional NLIDB, like the ones presented in [3] and it had a fast implementation – in 6 months it was answering simple questions. However, it soon became difficult to extend mainly due to the strong dependency between syntactic and semantic rules, the recursive organization of rules, and the need of complete analyses.

Having the above problems in mind, a new approach was followed [4] and a general QA system was built[5]. As in [6], we focused on a high-quality deep linguistic analysis of the question, made by a robust Natural Language Processing (NLP) chain. When applied to a restricted domain, a conceptual-semantic interpretation based on the domain terms is made, as suggested in [7].

In order to test our system in a restricted domain, we built a cinema database, based on information from both IMDB[4] and OSCAR.com[5]. It had the additional advantage of being a sympathetic application, simplifying the problem of finding users to test it. Moreover, although the restricted domain, questions could be complex enough to make question understanding a challenging task.

Having ready the whole NLP chain of the QA system, the first problem arise with the huge amount of names that can be asked about (more than 2. 500.000 movies, actors and staff names). As in order to correctly understand the question, those entities have to be precisely identified, our problem was to be able to make a good Named Entity Recognition (NER).

In a first attempt to solve the problem, we added actors and films names to the linguistic knowledge sources of our NLIDB. Unfortunately, that solution was unsustainable as the system became much slower. So we came out with another solution: to make a direct access to the database during the NLP chain, and to use the database itself to NER. In this paper, we present this approach and analyse its pros and cons.

We begin with a brief description of the whole NLIDB in section 2; then, we detail our NER technique (section 3) and we also evaluate it (section 4); finally, section 5 presents conclusions and future work.

2 General Architecture

Figure 1 shows the general architecture of the system.

We briefly describe the database in subsection 2.1; then in section 2.2 we present a short overview of the question deep linguistic analysis. Section 3 is dedicated to NER.

2.1 The database

Data is previously stored in a single database with 11 tables containing information about cinema acquired through IMDB and OSCAR.com. Data from

[4] Internet Movie Database (`http://www.imdb.com/`)
[5] The Oscars (`http://www.oscars.org/awardsdatabase/`)

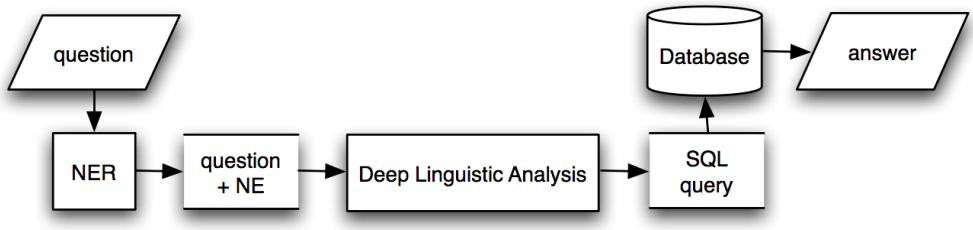

Fig. 1. General Architecture.

IMDB is available in several plain text files which are processed and stored in the database. As the information about the oscar awards was not available in the plain text files, it had to be acquired through a different source, and we took it directly from OSCAR.com. Notice that not all data from IMDB and OSCAR.com was considered, only the most important such as film cast, biographical information and the main oscar awards. As we will see in section 4, tests proved that most of the questions were related to this information.

To conclude, we highlight here the fact that full-text indexes in tables *persons* and *films* are an important feature as it allows full-text queries against those tables, which are the key for NER.

2.2 Question deep linguistic analysis

As said before, the NLIDB makes high-quality deep linguistic analysis of the question, through a robust NLP chain, that takes a question in natural language as input (as well as the possible named entities) and maps it into a SQL form (Figure 2).

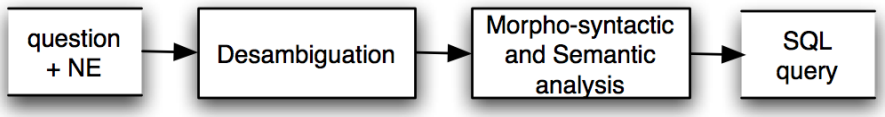

Fig. 2. NLP chain

These modules are briefly described in the following.

Disambiguation When we ask who is the director of "King Kong", we probably have in mind the latest film by Peter Jackson. Nevertheless, there's more than

one “King Kong” to be considered. In fact, to be accurate, four have the original title “King Kong”, and other two, from Japan, are also known as (A.K.A) “King Kong”. Assuming that the user requires the most popular film, we can believe that the latest film is the most popular one. However, there are also two films called “Psycho”: the original by Alfred Hitchcock from 1960 and a remake by Gus van Sant from 1998. Is the latest the most popular?

Given this, to allow the user to choose the film he actually means, after the named entities are recognized, the user can choose between two or more films with the same title (or between two or more people with the same name). To help the user's decision, we provide information about the film's opening year as well as the main cast. When it comes to people's name, the films where they stared are shown.

After this, even if there was nothing to disambiguate, the unique identifier (ID) of each film and/or person chosen is kept by the system. This information will allow a better performance in querying the database.

Morpho-syntactic and semantic analysis A dependency parser is responsible for the syntactic/semantic analysis and it outputs a XML file with the relevant information about the sentence. For instance, if we ask “Who is the director of Forrest Gump?” we want to retrieve a person's name based on the argument “Forrest Gump”. This argument is extracted from the sentence and a frame is generated:

```
<DEPENDENCY name="TARGET_WHO_DIRECTED">
       <PARAMETER ind="0" num="18" word="forrest gump"/>
</DEPENDENCY>
```

If we have a more complex question such as “Who acts with Clint Eastwood in Unforgiven?”, we would need two arguments to find the answer, therefore, the frame would be the following:

```
<DEPENDENCY name="TARGET_WHO_ACTS_WITH_IN">
     <PARAMETER ind="0" num="18" word="clint eastwood"/>
     <PARAMETER ind="1" num="19" word="unforgiven"/>
</DEPENDENCY>
```

These frames are then processed and a script that handles the database access is called. Considering the previous example, the script would get as arguments “clint eastwood” and “unforgiven”.

3 Named Entities Recognition

3.1 Overall process

The query NER is based on full-text queries, a feature provided in the later versions of MySQL that allows a natural language search through a column

in a database table. To perform these type of questions, the tables have to be previously built according to the code presented at the end of this section, where technical details can be found.

The syntax of a full-text search query is quite simple, for example, if we wanted to query the films table with the question "Who directed the film Forrest Gump?" we would perform this query:

```
SELECT * FROM films WHERE MATCH(title) AGAINST
                          ("Who directed the film Forrest Gump?");
```

We would get 1437 rows, ordered by greater relevance (10 first rows on table 1).

ID	TITLE
117754	forrest gump
185257	through the eyes of forrest gump
214266	black forrest gump
703433	die welt des forrest gump
336527	gump fiction
308842	foreskin gump
136207	andy gump for president
767193	directed by almodovar
449638	miss directed
667012	directed by andrei tarkovsky

Table 1. "Who directed the film Forrest Gump?" against table "films".

The longest match As we can see, the first match was the title "Forrest Gump", so we can set that part of the phrase as "film title". After this, we would perform the same search against the "persons" table and get a total of 380 rows. In table 2 we can see the 10 first results plus the 24th:

The 10 most relevant rows are not exact matches against the sentence. That must be a requirement, otherwise we would be recognizing a named entity that isn't in the question. Therefore, the only matched name would be the 24th row, namely "Forrest". As we had matched "Forrest Gump" before, it doesn't make sense to keep "Forrest" as it is not the longest match. As a result, the only named entity recognized for the sentence "Who directed the film Forrest Gump?" would be "Forrest Gump".

Let's now look at an example that will show that we can't just choose the longest name: "Which character is played by John Malkovich in Being John Malkovich?". We would match "Being John Malkovich" as film and then John Malkovich as a person, but "Being John Malkovich" is the longest phrase. We can't throw away "John Malkovich" in this case so, if a certain name or title appears more than once in the sentence, it is kept as a named entity.

ID	TITLE
1396185	h.s. gump
1396186	irving gump
1216100	diane gump
221580	richard b. gump
1216099	brandy gump
452670	andy gump
461279	eugene gump
1396184	david gump
174886	film
382417	dinar film
...	...
1341496	forrest

Table 2. "Who directed the film Forrest Gump?" against table "persons".

Narrowing the search Because we are dealing with a huge amount of data (about 800,000 rows in table "films" and 2 million rows in table "persons") this type of queries returns many rows. The sentence "Who directed the film Forrest Gump", for example, returned 1437 films. To avoid doing so many comparisons, and also because within the most relevant results lies the entities we want to recognize, we limit the query to return 100 rows and we try to match each one with the question. We must do so because we can have more than one person name and more than one film title in the sentence. For example: "In which film did John Malkovich and Glenn Close act together?" should have two named entities recognized: "Glenn Close" and "John Malkovich". When we perform this query we get 27567 results, but let us look at the first 10 in table 3.

ID	TITLE
251779	glenn close
531180	john malkovich
1204734	john close
30760	act four
382482	suhetu films
260391	films combo
382476	kolath films
390506	tripp films
382484	zojo films
224855	films catoosa

Table 3. "In which film did John Malkovich and Glenn Close act together?" against table "persons"

The first two are exact matches within the sentence. This means that we have to process every single row returned by the query because there could be

most than one match. In this case, both names "John Malkovich" and "Glenn Close" would be recognized.

Technical details The following is the SQL code required to use the full-text search feature in tables "persons" and "films":

```
create table if not exists imdb.films (
id integer unsigned auto_increment primary key,
title varchar(255) not null,
FULLTEXT (title),
);

create table if not exists imdb.persons (
id integer unsigned auto_increment primary key,
name varchar(255) not null,
FULLTEXT(name),
);
```

To perform full-text queries — natural language queries — the indexes must be previously built. This is done by adding "FULLTEXT(<column name>)" when the table is first created or it can be done afterwards through the ALTER TABLE command.

3.2 Major problems

ORDER	TITLE
1	directed by almodovar
2	miss directed
3	directed by john ford
4	the mis-directed kiss
5	directed by andrei tarkovsky
6	directed by alan smithee
7	directed by william wyler
8	directed by jacques tourneur
9	directed by norman foster
10	lisa's lunchbox directed
...	...
187	dangerous liaisions

Table 4. Ten first results for sentence "Who directed Dangerous Liaisions"

ORDER	TITLE
1	even more dangerous
2	dangerous
3	the dangerous
4	dangerous to know
5	down, out & dangerous
6	that dangerous age
7	a dangerous flirtation
8	dangerous women
9	she was a dangerous girl
10	the dangerous dude
...	...
157	dangerous liaisons

Table 5. Ten first results for phrase "Dangerous Liaisions"

The bigger problem of this technique is the need for total correctness. This means that the people names and films titles mentioned in the sentence must be written exactly as they are in the database. This demands total accuracy

from the user, which is sometimes difficult, specially if one is asking about a person/film in a language different from his native one. In table 4 we can see what would happen if one asks "Who directed Dangerous Liaisions" when the title is "Dangerous Liaisons".

It would be expected that "Dangerous Liaisons" was the first, or at least, at the top ten. Instead, as we can see in table 4 there isn't any title even close to it. In fact, "Dangerous Liaisions" is the 187th found by the query. We can see that the word "directed" has a great influence in the query so that all 10 first results have it within the title.

We could remove some keywords from the sentence to improve its accuracy on the title itself. In table 5 are the results just for "dangerous liaisions". Because there is a great resemblance between "dangerous liaisions" and "dangerous liaisons", in fact, the difference is just the extra "i", again we would expect that something very close would show up at the top, but it does not. In fact, it is the 157th row.

Even if the keyword removal improved the full-text search significantly, it would be risky to do so. We can't guess which words are part of a name or title and which aren't and we could be removing words from titles jeopardizing NER. Therefore, the best option is performing exact matches.

4 Evaluation

4.1 Experiments and results

The human-machine interface for this system is a plain web page. All it takes is writing the question in a text box and pressing the "submit" button.

During two weeks, 20 users performed a total of 300 questions and every single question and its answer was kept in a database. When analysing those questions, we could see that most of the questions were about film casts, oscar awards and biographical information and that our decision of only using part of the available information from IMDB did not affect its performance.

Let's now focus on the system's accuracy. The questions whose answer was "Couldn't understand the question. Please, try again" could not be answered due to:

- Unability to understand the question;
- Misrecognition of named entities.

Some questions are about information that just isn't available in the database, like "Where was Titanic shot?", and others, because of their complexity, can not be processed. For example: "Which actor stared in Magnolia and Top Gun?" is a complex SQL query that takes more than five seconds (an answer should not take more than that, otherwise the user quits asking questions).

This system's performance can be continuously improved, it is just a matter of adding new types of questions to it. For example, if we ask "Who is George Clooney" we will get all sorts of information: real name, nick name, date and

place of birth and latest films. If we ask "Where was George Clooney born?", the system can't understand the question because it was not yet added its processing.

From a total of 300 questions, 147 were answered and 153 couldn't be understood. We will analyse the ones that couldn't be understood or incorrectly answered because of a misrecognition of the named entities.

Notice, however, that as long as people names and films titles are typed correctly, NER through a database is 100% accurate.

4.2 Examples of misunderstood questions

The demand for total accuracy in name spelling can give the user the wrong idea about the system. We will show examples in the following.

- *Quantos filmes realizou joão cesar monteiro?* (How many films did joão cesar monteiro direct?)

This system answered "Cesar Monteiro: None" because it couldn't recognize the whole name. Cesar Monteiro, a person's name in the database, was not who the user wanted to know about, João César Monteiro, a Portuguese director, was the target of the question. As the user wrote "Cesar" instead of "César", the system could just match "Cesar Monteiro" giving answer about someone else. This innacuracy may take away the user's trust in the system.

- *Quem é Woody Alen?* (Who is Woody Alen?)

In the previous question, "Alen" is mispelled. Because of that, it is not possible to identify the famous actor/director Woody Allen. There are several "Woody's" in the database so the system can match just Woody. This can be quite frustrating for the user because none of the "Woody's" presented is the one he actually meant.

- *Em que filmes participou Dan Ackroyd?* (What movies was Dan Ackroyd in?)

Again, the person's name is mispelled so the system wasn't able to answer the question. The answer is "couldn't understand the question, try again.", and the user thinks that system can't answer that type of question while the problem is, in fact, the mispelling of "Dan Aykroyd".

Another problem, is the amount of films with similar titles that can lead to some confusion. Let's look at the following question to show it:

- *Quem contracena com Hugo Weaving in The Lord of the Rings* (Who acts with Hugo Weaving in The Lord of the Rings?)

The answer to this question was "Hugo Weaving isn't part of The Lord of the Rings cast" which sounds incorrect to the user. Actually, it is not. In fact, Hugo Weaving isn't part of the original "The Lord of the Rings" from 1978, though he

is part of the trilogy "The Lord of the Rings" from Peter Jackson: "The Lord of the Rings: The Fellowship of the Ring", "The Lord of the Rings: The Return of the King" and "The Lord of the Rings: The Two Towers". The problem is that people know the film just by the shortest version and assume the system will answer for the latest (and more popular) titles. That doesn't happen because the system performs an exact match, and "The Lord of the Rings" from 1978 will be the one it searches for. This is a one-time situation, but is still worth mentioning because such situation has a great influence on the user's opinion on the system.

5 Conclusion and Future Work

We have presented a technique to NER, based on full-text queries, a feature provided in the later versions of MySQL, that allows a natural language search through a column in a database table. We have shown that this method provides recognition of person names and film titles with little effort. In fact, as long as the names of the person and the films are typed correctly, the NER through a database is 100% accurate. Although it is a simple idea, it can be very usefull to NLIDB developers.

As future work, we will continue to increase the type of questions supported by the system, as well as the quality of the answers provided if there is any misunderstanding. Nevertheless, the first step will be toward the integration of an ortographic corrector. By doing this, we will be able to detect any NE, even if mispelled.

References

1. Mollá, D., González, J.L.V.: Question answering in restricted domains: An overview. Computational Linguistics **33**(1) (2007) 41–61
2. Authors: Authors title. (1997)
3. Jurafsky, D., Martin, J.H.: Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition. Prentice Hall PTR, Upper Saddle River, NJ, USA (2000)
4. Authors: Authors title. (2004)
5. Authors: Authors title. (2007)
6. Amaral, C., Figueira, H., Martins, A., Mendes, A., Mendes, P., Pinto, C.: Priberam's question answering system for portuguese. In Peters, C., Gey, F.C., Gonzalo, J., Müller, H., Jones, G.J.F., Kluck, M., Magnini, B., de Rijke, M., eds.: CLEF. Volume 4022 of Lecture Notes in Computer Science., Springer (2005) 410–419
7. Frank, A., Krieger, H.U., Xu, F., Uszkoreit, H., Crysmann, B., Jörg, B., Schäfer, U.: Querying structured knowledge sources. In: Proceedings of AAAI-05. Workshop on Question Answering in Restricted Domains, Pittsburgh, Pennsylvania (7 2005) 10–19