Addressing ontology-based question answering with collections of user queries

Óscar Ferrández *, Rubén Izquierdo, Sergio Ferrández, José Luis Vicedo

Natural Language Processing and Information Systems Group, Department of Software and Computing Systems, University of Alicante, E-03080 Alicante, Spain

A R T I C L E   I N F O

Article history:
Received 27 March 2008
Received in revised form 2 September 2008
Accepted 6 September 2008
Available online 31 October 2008

Keywords:
Question answering
Textual entailment
Natural language interfaces
Ontology modelling

A B S T R A C T

This paper presents QACID an ontology-based Question Answering system applied to the Cinema Domain. This system allows users to retrieve information from formal ontologies by using as input queries formulated in natural language. The original characteristic of QACID is the strategy used to fill the gap between users’ expressiveness and formal knowledge representation. This approach is based on collections of user queries and offers a simple adaptability to deal with multilingual capabilities, inter-domain portability and changes in user information requirements. All these capabilities permit developing Question Answering applications for actual users. This system has been developed and tested on the Spanish language and using an ontology modelling the cinema domain. The performance level achieved enables the use of the system in real environments.

1. Introduction

Question Answering (QA) appears with the aim of retrieving information required by natural language users’ queries. The purpose of a QA system is to find the correct answers to user arbitrary questions in both non-structured and structured collection of data. In the case of an ontology-based QA system, the data, where answers are sought, has a structured organization defined by an ontology.

An ontology formally defines a conceptual representation of concepts and their relationships within a specific domain. Ontologies play an essential role in the semantic web by enabling knowledge sharing and exchange. However, as stated in Bernstein, Kaufmann, Göhring, and Kiefer (2005), enabling non-expert users to exploit this information means bridging the gap between the logic-based semantic representation of the information and the natural language expressiveness employed by users.

Although for some time now there has been investigation of natural language interface systems (e.g. the LUNAR (Woods, 1970) and BASEBALL (Wolf, Chomksy, & Laughery, 1963) systems), approaches for real-world users focusing on formalized knowledge bases are still unavailable. As mentioned in Li, Yang, and Jagadish (2005), the two major obstacles are related with (1) understanding and representing natural language in a formal way and (2) translating this formal representation into a correct query adapted to the formal underlying ontology schema. On one hand, although research in Natural Language Processing (NLP) has been extensive, natural language comprehension still remains an open issue. On the other hand, even with...
the correct interpretation and representation of natural language, its translation over a formal ontology schema needs a suitable mapping between the free and open world of users’ words and expressions and the reduced and constrained vocabulary ontologies deal with.

Aimed at developing a methodology for building real applicable ontology-based QA systems, this work tackles the above mentioned problems from a perspective based on exploiting the different ways users ask for information in the domain.

There are two core elements in our approach: the user query formulation database and the entailment-based engine. Firstly, the system collects user queries from a given domain. These queries are then analyzed and grouped automatically into clusters according to several criteria where each cluster contains different expressions asking for the same information in the domain. These clusters are manually associated to a SPARQL expression which enables accessing the information required by the queries in the cluster. This database (a.k.a. user query formulation database) models the way of collecting queries about the given domain from users. Secondly, natural language questions are processed by the entailment-based engine. This engine infers semantic deductions between a new query and the clusters in the user query formulation database in order to associate this query to its corresponding SPARQL expression.

The rest of the paper is as follows. The next section presents our system at a glance. Section 3 describes in detail the creation of the user query formulation database. Section 4 delves into the entailment-based engine, which is the base of all the system’s inferences. Then, Section 5 presents the experiments carried out to evaluate this approach. Section 6 describes some related work and finally, Section 7 sums up the paper with our conclusions and future work proposals.

2. System overview

This section presents the architecture and functionality of QACID, a user-centred ontology-based Question Answering system. QACID system philosophy relies on five main components: the ontology, the data, the lexicon, the collections of user queries and the entailment engine. As it will be shown later, this approach has been applied and tested on Spanish language and using an ontology modelling the cinema domain.

- The ontology. Ontology-based Question Answering needs a formal representation of the information in the domain. Consequently, the ontology is one of the main entries to the system. To illustrate and test the system we consider an ontology which characterizes the key aspects of the cinema world. For this purpose we have used the OWL ontology designed for the tourism domain under the QALL-ME project (described in http://qallme.itc.it/) by restricting the whole model to only the classes and relations regarding the cinema domain. Fig. 1 illustrates the part of the ontology that contains the main relations and classes of the cinema domain that the system deals with.

---

1 SPARQL is a standardized and formal query language to retrieve information from RDF data (http://www.w3.org/TR/rdf-sparql-query).
2 The Ontology Web Language (OWL) is designed to be used by applications that need to process the content of information instead of just presenting information to humans http://www.w3.org/TR/owl-features/.
The data. The ontology has been populated with information about the domain. This information has been provided by LaNetro, a company that provides leisure guides and updated tourism information all over Spain (i.e. cinema, museums, gastronomy, hotels, restaurants, etc.). The data was stored in RDF format according to the ontology model. This RDF data is used as a structured database for extracting answers by means of SPARQL queries. In this implementation only information about movies and cinemas has been selected to populate our domain ontology.

The Lexicon. It is designed for establishing a mapping between words in natural language queries and ontology instances. Entities in the ontology, including classes, properties and instances, are directly extracted from the data and stored for fast access and matching. This way the Lexicon comprises the whole set of the ontology instances along with their respective ontology classes or property tags (for instance: “Casino Royale” ⟷ [MOVIE]; “Alicante” ⟷ [DESTINATION]; “Cinesa Panoramis” ⟷ [CINEMA]).

Collections of user queries. Instead of trying to guess the different ways users would ask the system for information, we decided to acquire all this automatically from users’ experiences. Keeping in mind the different kinds of data the ontology contains, a group of people were asked to formulate queries according to their respective interests in the cinema domain. These queries were automatically processed in order to derive a representative set of query patterns which define users’ requirements. These patterns were then associated with their corresponding SPARQL queries which permit accessing the information required in the question from the RDF database. As a result we obtain a set of pairs ⟨question pattern, SPARQL query⟩ representing what we know as user query formulation database.

The entailment engine. The core of the system is based on an entailment engine. This module uses entailment techniques to infer semantic deductions between a user query and the query patterns included in the user query formulation database previously obtained. This process allows the system to associate new incoming queries with their corresponding SPARQL expressions in order to retrieve the answer sought from the RDF database.

Fig. 2 depicts the overall system architecture. The QACID system takes as input the domain ontology populated with domain data, the user query formulation database and an informal query expressed in natural language. As output, the information required in the query is returned to the user.

The whole QA process is composed of two consecutive phases: question analysis and answer retrieval.

---

4 The Resource Description Framework (RDF) model. RDF is a set of specifications originally designed as a metadata model which is considered as a general method of modelling information http://www.w3.org/RDF.
The main aim of the *user query formulation database* is to have a representative and significant sample of queries reflecting users’ interests and needs focused on our representation of the cinema domain. The process of building the *user query formulation database* involves three consecutive steps:

- The first is to generate a set of representative queries according to the ontology domain. To do this, 50 Spanish speakers of different age, gender and nationality were selected to query our representation of the cinema domain. The ontology, together with a list of real entities extracted from our data (e.g. Cinema name, Movie name, ...) were shown to them in order to generate queries asking for any kind of data they were interested in. These queries contained real data instances for the concepts defined in the ontology (for example, an instance of “CINEMA” is “Abaco 3D”). This process compiled a set of 500 real free form queries extracted from the cinema domain (example in Table 1 is one of them). Although the users had available the ontology as well as the entity instances of our Lexicon, the way of asking for them was typical of each user.

- The second step consists of identifying and tagging the named entities appearing in this set of queries. The Entity Annotator module is applied to the 500 queries obtained in order to detect and classify the entity instances, replacing them by their corresponding ontology concept. For instance, the query “¿Dónde puedo ver Saw 3? – Where can I see Saw 3?” is annotated as “¿Dónde puedo ver [MOVIE]? – Where can I see [MOVIE]?”. As mentioned previously, the Entity Annotator module is based on fuzzy matching, trying to match strings in natural language with the entries of our lexicon. Once this process is completed, we remove all repeated queries obtaining a set of 348 distinct concept tagged queries.

- Finally, the set of distinct queries is automatically processed and grouped into clusters according to their semantic equivalences. Two queries are considered semantically equivalent when they represent the same knowledge scenario, meaning both queries ask for the same information and provide the same ontological concepts. For instance, the queries:

  - “¿Cuál es el número de teléfono del cine [CINEMA]?”
  - “¿Dónde puedo ver [MOVIE]? – Where can I see [MOVIE]?”

Table 1

<table>
<thead>
<tr>
<th>1 Informal input query</th>
<th>¿Dónde puedo ver Casino Royale? (Where can I see Casino Royale?)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 Morphological Analysis</td>
<td>¿Dónde puedo ver Casino Royale? (Where can I see Casino Royale?)</td>
</tr>
<tr>
<td>3 Tagging the query with ontology concepts</td>
<td>¿Dónde puedo ver [MOVIE]? (Where can I see [MOVIE]?)</td>
</tr>
<tr>
<td>4 SPARQL returned by the Entailment Engine</td>
<td>WHERE ?movie name [MOVIE], ?event hasEvent ?movie, ?event isInSite ?cinema, ?cinema name ?nameCinema</td>
</tr>
<tr>
<td>5 Incorporating the original input query entities into the SPARQL</td>
<td>SELECT DISTINCT ?nameCinema</td>
</tr>
<tr>
<td></td>
<td>WHERE (?movie name “Casino Royale”, ?event hasEvent ?movie, ?event isInSite ?cinema, ?cinema name ?nameCinema)</td>
</tr>
</tbody>
</table>

The next two sections explain in detail the creation of the *user query formulation database* and the entailment-based retrieval engine.

### 3. User query formulation database creation

The main aim of the *user query formulation database* is to have a representative and significant sample of queries reflecting users’ interests and needs focused on our representation of the cinema domain. The process of building the *user query formulation database* involves three consecutive steps:

- The first is to generate a set of representative queries according to the ontology domain. To do this, 50 Spanish speakers of different age, gender and nationality were selected to query our representation of the cinema domain. The ontology, together with a list of real entities extracted from our data (e.g. Cinema name, Movie name, ...) were shown to them in order to generate queries asking for any kind of data they were interested in. These queries contained real data instances for the concepts defined in the ontology (for example, an instance of “CINEMA” is “Abaco 3D”). This process compiled a set of 500 real free form queries extracted from the cinema domain (example in Table 1 is one of them). Although the users had available the ontology as well as the entity instances of our Lexicon, the way of asking for them was typical of each user.

- The second step consists of identifying and tagging the named entities appearing in this set of queries. The Entity Annotator module is applied to the 500 queries obtained in order to detect and classify the entity instances, replacing them by their corresponding ontology concept. For instance, the query “¿Dónde puedo ver Saw 3? – Where can I see Saw 3?” is annotated as “¿Dónde puedo ver [MOVIE]? – Where can I see [MOVIE]?”. As mentioned previously, the Entity Annotator module is based on fuzzy matching, trying to match strings in natural language with the entries of our lexicon. Once this process is completed, we remove all repeated queries obtaining a set of 348 distinct concept tagged queries.

- Finally, the set of distinct queries is automatically processed and grouped into clusters according to their semantic equivalences. Two queries are considered semantically equivalent when they represent the same knowledge scenario, meaning both queries ask for the same information and provide the same ontological concepts. For instance, the queries:

  - “¿Cuál es el número de teléfono del cine [CINEMA]?”
  - “¿Dónde puedo ver [MOVIE]? – Where can I see [MOVIE]?”

To overcome this task, we use the Freeeling toolkit [http://garraf.epsevg.upc.es/freeeling/](http://garraf.epsevg.upc.es/freeeling/).

“¿Cuál es el teléfono de contacto del cine [CINEMA]?”
“What is the contact number of the cinema [CINEMA]?”

belong to the same semantic cluster. Having processed the set of distinct queries, a total of 54 semantic clusters were obtained. Once the queries are semantically grouped, we associate a SPARQL expression to each cluster. This expression will enable retrieving the answer for any of the queries in the cluster. The SPARQLs contain generic concept labels (e.g. CINEMA) regardless of the data instances. Therefore, one should note that before launching the SPARQL over the RDF database, these labels will be replaced by the original data instances identified in the natural language query. When a semantic cluster is processed, we have several queries asking for the same information and it is very easy to extract knowledge about the kind of information users ask for. To take advantage of this, we have developed a semantic resource called the ontology attributes characterization. This characterization consists of knowing the different ways that the users ask for a specific ontology attribute. For instance, considering the queries belonging to the cluster of the previous example, we found three ways to request the telephone number attribute:

(i) número de teléfono (telephone number).
(ii) número de contacto (contact number).
(iii) número telefónico (telephonic number).

Thus, we extracted and stored the different ways of mentioning each ontology attribute in a query. This knowledge will be very useful for the entailment engine in order to detect paraphrases regarding the attributes that appear in the users’ queries.

Fig. 3 depicts the working flow of the user query formulation database creation.

To summarise, the user query formulation database is finally made up of 54 clusters, each one containing an average of 6.44 semantically equivalent queries, their corresponding SPARQL expression that defines the way of exploiting the RDF database, and the characterization of the attributes in the ontology. This characterization contains the different ways users have employed to refer to each attribute in the ontology.

The generation of this knowledge database has been built from users’ queries focused on the modelling of the domain represented by the ontology. Almost all of the ontology attributes and relations (88%) were covered by the final set of semantic clusters. However, for instance there were no queries asking for the e-mail, fax or GPS coordinates of a cinema. This acceptable recall proves that the methodology of building the user query formulation database has been appropriate according to the ontology modelling proposed, and the attributes without queries are surely less important for the requested users. One should note that the building methodology of the user query formulation database depends on the coverage of the ontology in modelling the domain as well as the variety of queries collected.

At this point, some discussion about other sources and methodologies to create the user query formulation database can be introduced. For instance, following the ideas exposed in (Chang, He, & Zhang, 2005) our semantic clusters could be derived from the Web, exploiting on-line information that satisfied the users’ needs. Additionally, repositories of on-line FAQ centered on specific domains could be used for this issue. However, in order to solve inconsistencies between changeable data we would need an Entity Annotator capable of detecting up-to-date and non-volatile data (e.g. within cinema domain it
would be useful to recognise very popular films (e.g. Casablanca), since we can find a lot of queries about them on the Web, and also identify movies now showing). Another drawback of these strategies is that they rely on the scarcity of information about this domain in Spanish. Nevertheless, this point opens a new line of further research orientated on how to complete and improve the user query formulation database from different sources (e.g. a complementary knowledge provided by Web mining engines could be added in order to enhance our cinema domain representation).

4. Entailment engine

It is well-known that human languages are extremely rich and ambiguous and a lot of applications in many NLP areas are highly influenced by the problem of language variability. In fact, an ambiguous text might represent several distinct meanings and a concrete meaning might be expressed in different ways. To address this language variability the concept of Textual Entailment has recently emerged. It has been defined as a generic framework for modelling semantic implications where a concrete meaning is described in different ways as proposed by Dagan and Glickman (2004).

In order to address the uncertainty that underlies textual entailment relations, the research community has proposed in recent years a wide range of approaches based on lexical (Ferrández, Micol, Muñoz, & Palomar, 2007c; Malakasiotis & Androutsopoulos, 2007), syntactic (Ferrández et al., 2007c; Kouylekov, 2006), semantic and logic models (Roth & Sammons, 2007) and of course approaches implemented by a combination of these (Hickl & Bensley, 2007; Iftene & Balahur-Dobrescu, 2007; Tatu & Moldovan, 2007; Zanzotto, Pennacchiotti, & Moschitti, 2007). In the case of this research, we have orientated the recognition of entailment relations towards the specific domain that we are dealing with. Whilst obvious that it has been inspired by our previous work (Ferrández, Micol, Muñoz, & Palomar, 2007a, 2007b, 2007c), to apply the new QA paradigm some adjustments and new features have been added to the system.

Our entailment-based QA core attempts to establish lexical-semantic inferences between a new input query and the predefined set of query patterns from the user query formulation database (see Section 3). This approach tackles the implications as one-way meaning relation between two queries, in which the meaning of one (the hypothesis or in our case the meaning of a pattern stored in the database) must be inferred by the meaning of the other (the text or in the case of this research a new query). This statement follows the methodology proposed in Glickman (2005) for entailment relations.

Aimed at achieving these goals and keeping in mind that the approach is going to be released as a web application, in the course of the system’s construction we have endeavoured to use as few as possible external resources. The approach implements several lexical implications and develops shallow semantic analyses focused on the knowledge provided by the ontology.

4.1. Lexical inferences

The lexical perspective relies on a wide range of similarity measures commonly used for textual entailment tasks, which are based on word co-occurrences as well as the context where they appear. The choice of these lexical measures was inspired by previous work in textual entailment recognition (Ferrández et al., 2007c; Malakasiotis & Androutsopoulos, 2007), where these measures are successfully applied to the task of detecting entailment relations.

To compute these measures, each query is considered as a bag-of-words without stop-words. It is part simplicity and part accuracy: it is less computationally expensive to compare terms between sentences than to implement the sentence structure and consume syntactic analysing tools. Moreover, systems using lexical inferences achieve similar performance results to systems employing deep language and logical-based representations (see Giampiccolo, Magnini, Dagan, & Dolan, 2007).

Next, we list the final set of lexical measures that have been used together with a brief description:

- **Smith–Waterman algorithm**: the Smith–Waterman algorithm is a well-known dynamic programming algorithm for performing local sequence alignment and determining similar regions between sequences. The algorithm was first proposed by Smith and Waterman (1981) and consists of two steps: (i) calculate the similarity matrix score; and (ii) according to the dynamic programming method, trace back the similarity matrix to search for the optimal alignment. For two sequences \( SQ_1 \) and \( SQ_2 \), the optimal alignment score of two sub-sequence \( SQ_1[i] \ldots SQ_1[1] \) and \( SQ_2[1] \ldots SQ_2[j] \) is the calculation of \( D(i,j) \) defined as

\[
D(i,j) = \begin{cases} 
0 & \text{start over,} \\
D(i-1,j-1) - f(SQ_1[i], SQ_2[j]) & \text{substitution or copy,} \\
D(i-1,j) - \text{GAP} & \text{insertion,} \\
D(i,j-1) - \text{GAP} & \text{deletion.}
\end{cases}
\]

The algorithm permits two adjustable parameters regarding substitutions and copies for an alphabet mapping (the \( f \) function) and also allows costs to be attributed to a GAP for insertions or deletions. In our experiments we empirically set the values 0.3, −1 and 2 for a gap, copy and substitution, respectively.

7 For some measures we have used implementations provided by SimMetrics library [www.dcs.shef.ac.uk/~sam/simmetrics.html](http://www.dcs.shef.ac.uk/~sam/simmetrics.html).
• **Consecutive subsequence matching:** this measure assigns the highest relevance to the appearance of consecutive subsequences. In order to perform this, we have generated all possible sets of consecutive subsequences, from length two until the length in words, from the text and the hypothesis. If we proceed as mentioned, the sets of length two extracted from the hypothesis will be compared to the sets of the same length from the text. If the same element is present in both the text and the hypothesis set, then a unit is added to the accumulated weight. This procedure is applied for all sets of different length extracted from the hypothesis. Finally, the sum of the weight obtained from each set of a specific length is normalized by the number of sets corresponding to this length, and the final accumulated weight is also normalized by the length of the hypothesis in words minus one. These equations detail this measure:

\[
C_{\text{match}} = \frac{\sum_{i=2}^{n} f(SH_i)}{|H| - 1},
\]

where \(SH_i\) contains the hypothesis’ subsequences of length \(i\), and \(f(SH_i)\) is defined as follows:

\[
f(SH_i) = \frac{\sum_{j=1}^{n} \text{match}(j)}{|H| - i + 1},
\]

being \(\text{match}(j)\) equal to one if there exists an element \(k\) that belongs to the set that contains the text’s subsequences of length \(i\), such that \(k = j\). We should like to point out that this measure does not consider non-consecutive subsequences. In addition, it assigns the same relevance to all consecutive subsequences with the same length. Furthermore, the longer the subsequence is, the more relevant it will be considered.

• **Jaro distance:** this metric comes from the work presented in Jaro (1989, 1995) and measures the similarity between two strings taking into account spelling derivations. The following equation describes how the similarities are obtained:

\[
d_{\text{Jaro}}(s_1, s_2) = \frac{m}{3 \cdot |s_1| + 3 \cdot |s_2| + m - t},
\]

being \(s_1\) and \(s_2\) the strings to be compared, \(|s_1|\) and \(|s_2|\) their respective lengths, \(m\) the number of matching characters considering only those not further than \(\frac{\text{max}(|s_1|, |s_2|)}{2} \) and \(t\) the number of transpositions computed as the number of matching (but different) characters divided by two.

• **Euclidean distance:** the traditional definition measures the distance between two points \(P = (p_1, p_2, \ldots, p_n)\) and \(Q = (q_1, q_2, \ldots, q_n)\) in Euclidean \(n\)-space as

\[
\sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \cdots + (p_n - q_n)^2} = \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2}.
\]

With the aim of dealing with strings, we set \(n\) as the number of distinct items in either of the two strings and \(p_i, q_i\) the times that each of them appears in each string, respectively.

• **Jaccard similarity coefficient:** is a statistic coefficient for comparing the similarity and diversity of sample sets. It is defined as the size of the intersection divided by the size of the union of the sample sets:

\[
J(A, B) = \frac{|A \cap B|}{|A \cup B|}.
\]

In our case, we compute this coefficient representing each string as a Jaccard vector. This metric was first introduced and detailed in Jaccard (1912).

In addition to the aforementioned lexical measures and owing to the idiosyncrasies of this novel framework context (i.e. entailment detection between queries), we have also introduced an inference corresponding to the interrogative query terms under the hypothesis that it would suitably support the entailment process.

• **Wh-terms:** or interrogative terms (e.g. when, what, where, etc.) play an important role within the queries’ meanings. Although the previous lexical measures take into account every token, including Wh-terms, these measures do not detect the importance of the presence or absence of these terms in both queries involved in the entailment process. For instance, a query asking for a place (Wh-term: where) semantically differs from a query asking for a period of time (Wh-term: when). We strongly believe that such a situation would be useful to determine the entailment. Hence, we added to the system a binary feature that integrates the capability of knowing when two queries have the same Wh-term. It helps the system to rank the candidates in the entailment decision process, but it is totally clueless as to determine the entailment by itself.

### 4.2. Semantic ontology-based inferences

Moreover, to make the system more accurate, apart from the lexical measures we have integrated some semantic knowledge directly derived from the ontology.
**Query concept constraint**: during the creation of the user query formulation database, the Entity Annotator module tagged all entity instances with their ontology concept. Having this information and the whole list of ontology concepts shown to the users, a constraint to be fulfilled by every entailment pair should be the following:

'The candidate entailment pair of queries must embed the same entities in number and type'

*Fig. 4* shows an input annotated query together with the candidates (in light grey) and non-candidates (in dark grey) patterns that could produce an entailment inference according to this constraint. This constraint throws out of the entailment process those patterns that do not contain the same number and entity types detected in the input query.

**Attribute-based inference**: taking advantage of the knowledge acquired regarding the different ways of naming the ontology attributes (i.e. the ontology attributes characterization, see Section 3), we have performed for each query an attribute-based inference about the information that the query is requesting. The procedure uses the ontology attributes characterization in order to detect the presence of the ontology attributes in the query (normally these attributes are the information solicited). Later on, this procedure positively weights those patterns that contain attributes equivalent to the ones within the input query. Two attributes are equivalents whether they are expressed in the same manner or using a paraphrase of them among the paraphrases stored in the ontology attributes characterization. The final weight obtained by this inference is defined as follows:

\[
Abl_{\text{weight}} = \frac{\sum_{a_i \in Q, a_j \in P} Eql(a_i, a_j)}{|Q|},
\]

where \(Q\) and \(P\) contain the attributes that appear in the input query and each database pattern respectively, and \(Eql(a_i, a_j)\) takes the following value:

\[
Eql(a_i, a_j) = \begin{cases} 
1 & \text{if } a_i = a_j \text{ or } a_i \text{ is a paraphrase of } a_j, \\
0 & \text{otherwise}. 
\end{cases}
\]

Therefore, if two or more attributes are found in the query, they are considered of equal importance. However, for patterns without the requested attribute (or any of its paraphrases) this weight will be set as zero.

Finally, each lexical and semantic inference obtains a similarity weight between zero and one, hence we calculate the final entailment coefficient as the sum of all weights divided by the number of inferences considered. To determine the entailment decision, we empirically establish a threshold over the patterns from the user query formulation database. Therefore, if the final entailment coefficients are higher than the threshold they are considered as true entailment deductions.

As we can observe, all the inferences contribute equally in the entailment score formula. It would have been desirable to apply any kind of machine learning classifier in order to weight adequately each inference according to its relative importance in the entailment process. However, at the current state of this research the shortage of training samples renders the application of these methods useless. As subsequent work we plan to enlarge and enrich the training collection by collecting the questions users are posing through the QACID system. This will enable applying machine learning techniques to assess the information gain that each inference provides to the final decision.

**5. System evaluation**

In order to assess the effectiveness of QACID, an evaluation framework over the cinema domain was created. Two main targets are intended to be covered by this evaluation. On one hand, due to the fact that the overall performance of our system...
is entirely influenced by the correct entailment detection, we created an evaluation focused on the capability of the entailment engine to detect correct inferences between queries (entailment evaluation). On the other hand, we set an evaluation destined for measuring our system behaviour when users, who are unacquainted with the structure and content of the system, pose queries (on-field evaluation).

5.1. Evaluation environments

With the two evaluation frameworks detailed below, we measure the effectiveness of the whole system, the entailment engine and how the representative is our user query formulation database.

- **Entailment evaluation**: for this purpose, 10 new users were asked to formulate one query for each semantic cluster using the instances stored in our entity lexicon. In total 540 new input queries were generated. These new queries were divided into two sets: the training and the test set. The former was used to adjust the entailment decision threshold, in such a way that when the similarity score returned by the entailment engine is lower than the predefined threshold this query is considered as an uncertain because there is not any pattern for which the entailment coefficient overcomes the threshold. Whereas the latter was used as a blind set to evaluate the final system performance. These sets comprise 378 (7 users) and 162 (3 users) queries respectively. This aside, one should note that within the new sets repeated queries have not been discarded, and all queries ask for information belonging to our domain. This means that this evaluation does not consider out-of-domain queries nor queries asking for unknown ontology concepts, properties or relations. In brief, this framework is intended for evaluating the performance of the entailment engine in the task of extracting the most similar semantic pattern.

- **On-field evaluation**: with the aim of assessing our system’s effectiveness in a real application scenario, 10 new users\(^8\) were requested to formulate 10 spontaneous and independently-generated queries about the cinema domain. These new users were recruited from non-research environments (e.g. high school students and administrative assistants amongst others), and they did not know anything about our system and research. There were no specific instructions given to these new users, they were only requested to make a query asking for any information about movies or cinemas in Spanish. This evaluation will show how the ontology fulfils the users’ needs in the cinema domain as well as the semantic recall of the system faced with totally unacquainted users.

5.2. Results analysis

Fig. 5 draws four graphics showing the evolution of the precision, recall and \(F\)-measure depending on the value of the entailment decision threshold and using our training query set. Each graphic corresponds to the experiments carried out taking into account the different inferences exposed in Sections 4.1 and 4.2:

- **Lexical baseline (LB)**: implements the whole set of basic lexical measures presented in Section 4.1, which have been successfully used in previous specialized research in textual entailment. This semantic-free experiment establishes our baseline in order to measure the improvement that occurs with the addition of the remaining inferences.

- **LB + Wh-terms inference (LB + Wh)**: adds to LB the inference about the interrogative terms that appear in the queries.

- **LB + Wh + concept constraint (LB + Wh + CC)**: adds to the lexical perspective the constraint regarding the correspondence between the ontology concepts.

- **LB + Wh + CC + attribute-based inference (LB + Wh + CC + AbI)**: develops all lexical and semantic inferences including the inference deduced by the paraphrase of ontological attributes.

As we can observe, the system’s precision keeps good rates for every threshold value due to the fact that almost all new queries entail some of the patterns from the database. This fact shows that users make their queries using similar expressions, and different users ask for the same information using the same wording. These situations are very well-managed by the system due to the building methodology of our user query formulation database.

According to the training query set, the threshold value that achieves the best precision without compromising the system recall is 0.7 for the LB (our baseline), 0.6 for the LB + Wh and LB + Wh + CC experiments and 0.5 for the LB + Wh + CC + AbI one. Similarity coefficients higher than these thresholds will be considered as entailment candidates, and the highest one will be returned by the system as the entailed pattern.

Table 2 illustrates the entailment evaluation results obtained over the set of blind queries showing the amount of queries tagged as correct (when the system responds correctly), incorrect (the system returns a wrong answer) or uncertain (when there is not any pattern for which the entailment coefficient overcomes the empirical decision threshold). As seen, although the percentages of precision, recall and \(F\)-measure are slightly lower than the ones obtained during the training stage (see Fig. 5), the system’s behaviour is somewhat similar for both the training and blind set.

---

\(^8\) These users do not know anything about our system ontology structure nor the data instances it contains.
As anticipated, the LB and LB + Wh experiments obtain the lowest results. Although the LB experiment achieves a slight increase in precision, due to the fact of its lack of knowledge the recall decreases dramatically leaving 39 queries without an entailment decision. Without doubt these experiments are significantly improved with the addition of semantic ontology-based knowledge. For instance, the simple constraint about the presence of the ontology concepts (LB + Wh + CC experiment) improved the $F$-measure by 12.91% with regards to the LB + Wh experiment, whereas the experiment considering the paraphrase alignment between ontological attributes (LB + Wh + CC + AbI) produced an improvement of 30.88%.

Finally, Table 3 shows the results obtained by performing the on-field evaluation.
From the whole set of 100 on-field evaluation queries, 80 queries were well-answered\(^9\) satisfying the users’ requirements (i.e. precision rate of 80\%), whereas for the rest of the queries the system returned wrong answers or instead, it was not able to give an answer (9 incorrect and 11 uncertain queries, respectively). Regarding the errors (incorrect and uncertain queries), we distinguish several kinds as listed below:

- **Out-of-coverage queries**: queries that require concepts, relations or properties not reflected within our ontology. For instance, the query “¿Cuánto cuestan las palomitas en el cine Colci?” (How much does popcorn cost at the Colci cinema?) asks for a property of the concept cinema that is not reflected in our ontology. Of the system’s total errors, 11 (55\%) queries were out of coverage, seven were uncertain because the system did not return a pattern, while the rest were answered wrongly. An ideal system should have marked all these queries as uncertain, since the information requested is not reflected in the ontology. However, in some cases the entailment engine infers wrong patterns owing to the fact that although the queries ask for unknown ontology concepts they contain instances of known concepts and the entailment threshold is surpassed.

- **Semantically unsupported queries**: queries asking for concepts, relations or properties of our ontology, but without correspondence with the semantic clusters generated. For instance, the query “¿De dónde es el correo electrónico del cine Astoria?” (What is the e-mail of the Astoria cinema?) solicits a cinema property reflected by the ontology, but there is not any semantic cluster derived from the user query formulation database (and consequently any query formulated by the users) capable of supporting this query. Only 3 (15\%) of the errors queries result in this kind of error: 2 returning wrong answers and 1 tagged as uncertain. The system behaviour is somewhat similar to the treatment of out-of-coverage queries, but in this case we can use these queries to extend the semantic recall of the system generated clusters.

- **Pure entailment error**: three times (15\%) it occurs that the query belongs to one of our semantic clusters, but the system returns a wrong answer. This we name as a pure entailment error. Future improvements in the entailment engine will help to prevent these situations occurring.

- **Verification queries**: during the creation of the user query formulation database, the users were requested to ask for concepts, relations or properties of our ontology. Following this methodology the system is unable to answer verification queries. However, within the on-field evaluation we can find 3 verification queries (e.g. “¿Están dando Casino Royale en el cine Colci?”, Is Casino Royale shown at the Colci cinema?) for which the system does not have associated patterns (i.e. tagged as uncertain queries). An extension design to control this kind of queries is also planned as a future improvement.

From a technical point of view these evaluations prove: (1) the correct application of the lexical measures that achieve good semantic-free similarity rates useful in many entailment cases; and (2) the complementary knowledge supplied by the semantic inferences based on the ontology. Whilst from a methodological view they show (3) a more than acceptable recall of the semantic clusters generated.

However, we should like to point out some examples of failure together with a possible system improvement that should solve them:

**1st example:**

Input Query (asking for a film studio):

“Dime dónde fue rodada la película [MOVIE].”

“Tell me where the film [MOVIE] was shot.”

Returned associated pattern (asking for the nationality of the film):

“¿De dónde es la película [MOVIE]?”

“Where is the film [MOVIE] from?”

**2nd example:**

Input Query (asking for a time period):

“¿Cuándo puedo ir a ver la película [MOVIE]?”

---

\(^9\) Well-answered queries are those that the users considered as correct, since these answers provided the information solicited.
“When can I go to see the film [MOVIE]?”

Returned associated pattern (asking for cinemas):
“¿Dónde puedo ir a ver la película [MOVIE]?”
“Where can I go to see the film [MOVIE]?”

Regarding the first example, the system is not able to distinguish that the input query is asking for “the studio where the film [MOVIE] was shot”, whereas the returned pattern solicits “the nationality of the film [MOVIE]”. This occurs because the queries use the same Wh-term and no attribute is mentioned within them. For instance, if the input query had been formulated as “Dime el estudio donde se ha rodado la película [MOVIE]” (“Tell me the studio where the film [MOVIE] has been shot”) the entailment engine would have returned a successful pattern.

In order to solve this deficiency we need a special treatment of certain verbs. For instance, if we establish that the verb “shoot” is associated with a film studio, and then we entail positively the queries that contain “shoot” with the patterns related to film studios, this drawback would be solved. Therefore, a study about the possible implications that the verbs of the domain could have over the ontology concepts/attributes would be very useful to improve the final entailment decision.

With regards to the second example, the wrong entailment inference is produced because both queries are similar except for the Wh-term. Our first impression to overcome such a mistake is to give more importance to the Wh-terms, but it might introduce a lot of new errors into the system. Another solution for this problem would consist of detecting the Expected Answer Type (EAT) of the input queries. Although to some extent it is inferred by means of the ontology attribute characterization, when the requested attribute does not appear in the query (as in the two aforementioned examples) there is no way to know what the query is asking for.

Hence, an EAT recognizer would solve the second failure example since only the patterns asking for the same EAT appearing in the input query would be considered as entailment candidates. In addition to this it should be noted that EAT recognition would actually help to solve the first mistake as well.

6. Related work

Natural language interfaces to databases have been extensively studied of late (Androutsopoulos, 1996; Androutsopoulos, Ritchie, & Thanisch, 1995; Chan & Lim, 2003; Copestake & Jones, 1990; Filipé & Mamede, 2000; Minock, 2005; Popescu, Etzioni, & Kautz, 2003). These kinds of tools allow users to formulate their requirements of information in natural language queries and retrieve the required information from knowledge bases. According to existing publications, there are two categories of natural language interfaces that define the ability of processing users informal queries:

(1) **Full natural language interfaces.** The system that implements this kind of interfaces, processes queries formulated in natural language without any terminology restrictions.

(2) **Restricted natural language interfaces.** The systems process queries that are formulated in a controlled language. In this case, users have to previously learn the characteristics of the query language to be able to formulate queries.

Our approach falls into the first category. In QACID, the users insert their information needs through full natural language queries.

We can find a typical example in Androutsopoulos, Ritchie, and Thanisch (1993), which processes the natural language query with the aim of transforming it into an intermediate logic representation to be later translated into SQL. Some other approaches (Thompson & Mooney, 1999; Zelle & Mooney, 1996; Zhang & Yu, 2001) employ machine learning methods for transforming informal queries into logic format representations. These systems are trained using specific domain data based on those provided by Mooney and obtain accuracy rates circa 90%. However, they need a lot of domain specific training data to achieve these scores. Some other approaches (Kang, Na, Lee, & Yang, 2004; Rodrigo et al., 2005) try to divide the query into words (keywords) and compose a formal query regarding the knowledge supplied by them. Another example is the system exposed in Li et al. (2005, 2006) which permits access to an XML12 database. Finally, we want to point out two systems presented in Popescu et al. (2003) and Wang et al. (2007) that translate the full natural language queries into SPARQL format.

The processing of natural language queries is often complex, fuzzy and ambiguous. For this reason many systems only process queries composed over a controlled language with terminology and grammatical restrictions. However, to use these systems the users have to previously learn the syntax of the controlled language in order to encapsulate the queries into it. The system presented in Bernstein et al. (2005) uses “Attempto Controlled English” in order to query ontologies. Another example Nelken and Francez (2000) also develops a controlled language with the aim of accessing historical databases. Following this line, Popescu et al. (2003) defines the notion of semantic tractable questions trying to anticipate the requirements a question has to fulfil in order to be successfully translated into a SQL query. However, in this approach questions with unknown words are not semantically tractable and cannot be processed.

---

10 SQL (Structured Query Language) is a database computer language designed for the retrieval and management of data in relational database management systems (RDBMS), database schema creation and modification, and database object access control management.


12 [http://www.w3.org/XML/](http://www.w3.org/XML/).
Pattern-based methods are also used to solve tasks focused on natural language interfaces. The system (Lopez, Pasin, & Motta, 2005, 2007) processes the queries classifying them into 23 categories. If the input query is classified into one of these categories, the system will process it correctly. However, the limited coverage of the patterns means many queries are left unresolved. In contrast, although QACID bases its implementation on the use of patterns, it solves this handicap using a textual entailment processing that lets it classify most input queries. Finally, we would like to show a system (Bernstein & Kaufmann, 2008) that helps users to form the query and avoids the ambiguity by means of a guided natural language search engine. However, once again the processing ability of this kind of system is limited to a controlled language.

7. Conclusions and future work

The main contribution of this research is the development of a methodology to create a restricted domain QA system based on collections of user queries. Specifically, we present QACID, a user-centred ontology-based QA system developed for answering queries related to the cinema domain. The system makes use of an ontology that represents the domain, the information modelled on RDF format and a pattern database generated from the users’ interests and needs. To retrieve answers for new queries, the system implements an entailment engine in order to deduct semantic inferences between them and the predefined users’ requirements.

Consequently, this methodology can be applied to other domains as well as other languages. For a new system working on a different domain, we only need an ontology modelling this concrete domain to generate the user query formulation database with representative sample queries (regardless whether we use an existing ontology or we create a new one). Besides, when making the system portable to other languages, we only need to create the user query formulation database for these specific languages. For this purpose, it would also be necessary to manually associate for each semantic cluster a SPARQL query to retrieve the required information.

Moreover, the proposed methodology also allows the system to be easily created and adapted to new information requirements in an incremental way. In order to automatically enhance the system’s semantic recall a user–machine dialogue could be established in order to encapsulate unknown queries into the predefined set of semantic cluster or to create new clusters to contain these queries. For instance, in the case of the on-field evaluation (see Section 5.2) the semantically unsupported queries could be used to extend the recall of the semantic clusters by creating new clusters. In addition, the out-of-coverage queries would be treated in order to incorporate new concepts, relations or properties within our ontology. This research line (to investigate the automatic extension of the system) is our next prime objective.

Results point out that accurate precision is reached by the entailment engine. More importantly, this precision has been achieved using a reduced number of semantic resources so as not to compromise the system’s speed. This fact makes its use feasible in an on-line application.

Throughout this work, we have identified several lines of research that sum up our future work proposals: (1) we will design a procedure to take advantage of the EAT detection in the input queries as well as the creation of semantic resources that make it feasible to detect inferences between the domain terminology (e.g. the verb “shoot” could be related to a film studio); (2) we plan to extend the system towards controlling verification queries and questions requiring some calculus as output (e.g. “How many cinemas are there in Spain?”); (3) complete and improve the user query formulation database by means of exploiting different kinds of sources (e.g. FAQ repositories, Web logs and Web mining techniques); and (4) enrich the set of training samples in order to make feasible the use of a machine learning classifier for optimizing the entailment engine.

Additionally, we are also investigating the inclusion of temporal and spatial context aware capabilities to the system. This will enable the system to answer queries such as “¿Dónde puedo ver Casino Royale mañana?” (“Where can I see Casino Royale tomorrow?”) or “¿Cuál es el cine más cercano donde puedo ver Casino Royale?” (“What is the nearest cinema where I can see Casino Royale?”). Such expertise will be added to our approach in the near future.

Finally, we would like to mention that a prototype-demo version of the system is available at http://sqm1.dlsi.ua.es/QACID/index.jsp in order to assess the performance of the system in its early stages.

References


