EQUAL: Encyclopaedic QUestion Answering for Lists

Iustin Dornescu
RIILP, University of Wolverhampton
i.dornescu2@wlv.ac.uk

Abstract

This paper presents *EQUAL*, a semantic question answering (QA) system which relies on structural information from Wikipedia to extract answers to list questions in the GikiCLEF 2009 task. The system is described in the context of of a more general architecture for semantic QA. Unlike the usual textual QA approach, *EQUAL* does not rely on identifying the answer within a text snippet by using keyword retrieval. Instead, it explores the Wikipedia page graph extracting and aggregating information from multiple documents. Enforcing semantic constraints proved to yield precise results, and the system achieved the highest score amongst the participants in the task.

Categories and Subject Descriptors

H.3 [**Information Storage and Retrieval**]: H.3.1 Content Analysis and Indexing; H.3.3 Information Search and Retrieval; H.3.4 Systems and Software; H.3.7 Digital Libraries; H.2.3 [**Database Managment**]: Languages—*Query Languages*

General Terms

Measurement, Performance, Experimentation

Keywords

Question answering, List questions, Wikipedia, Semantic question answering

1 Motivation

This paper describes the *EQUAL* system and its participation in GikiCLEF[1] 2009. The system was designed to combine the coverage of open-domain textual QA with the precision/accuracy and semantics of closed-domain QA, whilst also addressing some of the limitations of standard textual QA systems. The feed-forward pipeline architecture often employed in QA exacerbates errors, mainly due to the use of approximate solutions to difficult NLP problems during intermediate steps. Systems usually rely on the hypothesis of information redundancy in large corpora, focusing on question rewriting patterns (e.g. interrogative vs. affirmative) and keyword expansion to identify the best paragraphs (snippets). This makes textual QA essentially a 'smart' paragraph retrieval system, enhanced with post-retrieval 'extraction' and weak semantic algorithms. Aggregating information from multiple documents usually means re-ranking

[1] http://www.linguateca.pt/GikiCLEF/

candidates based on the extraction frequency of each one. Statistical prevalence is a surrogate of real semantic information which, in many QA systems, is discarded in favour of the distributional hypothesis: "similar words appear in similar contexts". The use of standard NLP tools also imposes a constraint regarding the content that is used in QA: plain text. The rich structural information common in HTML pages (e.g. links, tables) and their (implicit) semantics are largely ignored. While using plain text has certain advantages, current NLP tools are not powerful enough to compensate for the loss of information when the mark-up is removed.

Although textual QA works very well for certain classes of questions, the techniques employed limit the range of "feasible" questions to those which can be directly answered by a textual snippet. When the information necessary to compile the answer is spread across several documents, traditional systems lack the semantic representations necessary to combine the different pieces of information. This is not only due to the difficulty of general purpose inference (an AI-hard problem) but also to the complexity of different NLP tasks. As new resources have become available, the time is now right to add more semantic processing to the QA task. Newer campaigns such as the GikiCLEF task or in the INEX Entity Ranking[2] task focus on list questions, which are known to be difficult for most textual QA systems.

This paper presents an alternative semantic QA architecture and prototype system *EQUAL* to address current limitations of textual QA. The rest of the paper is structured as follows: Section 2 outlines the general architecture of semantic QA. Section 3 describes the way natural language is converted into semantic representations and Section 4 shows how answers are retrieved for these question interpretations. Section 5 discusses the results and related work, and the paper finishes with conclusions and ideas for future work.

2 Semantic QA Architecture

To successfully address list questions, a novel architecture is proposed that does not have words at its core, but entities. The semantic QA architecture, of which *EQUAL* is a prototype, has two main processing phases: a) the **analysis** phase, which is responsible for understanding the question and identifying answers, and b) the **feedback** phase, responsible for interacting with the user. The purpose of the latter is not only user modelling and dialogue management, but also the efficient interface between the human and the machine, allowing clarification questions for disambiguation, named sets of results for subsequent querying or presenting additional information[3]. This paper focuses on the analysis phase.

One of the key ideas behind the semantic QA architecture is its ability to address ambiguity. Rather than employing standard NLP tools to get a most probable output, the system must be aware of different sources of ambiguity and be able to deal with them directly during processing. Humans use common sense and their knowledge of the world when understanding and interpreting questions. To compensate for the lack of such abilities, during the analysis phase the system must generate all possible interpretations and rank them based on the results found and interaction with the user.

All questions are interpreted as natural language retrieval queries. Therefore each question needs to be parsed and the resulting semantic structures need to be converted into formal queries which will correspond to a retrieval algorithm. The two main issues addressed in this paper are the mapping of natural language to formal semantic constituents (the semantic interpretation of the question) and the particular data model chosen to represent information according to the semantic model (retrieving the answers). The third main issue (not addressed here) is the generation of metadata for feedback and interaction.

The data model adopted is closely related to *Datalog* and *RDF*[4]. Each entity corresponds to a node in a graph. It has attributes, similar to *owl:DatatypeProperties*, and relations with other nodes, corresponding to *owl:ObjectProperty*. This abstract model facilitates the aggregation of information from various sources as needed at query time. For example, instead of having *Paris* as an entry in the *LOCATION* gazetteer, this model allows a QA system to "know" that *Paris* is a Roman establishment and a French city, to "know"

[2] http://www.inex.otago.ac.nz/tracks/entity-ranking/entity-ranking.asp

[3] The feedback phase is beyond the scope of the current paper, given that GikiCLEF is not an interactive task. However, it is a major component of the semantic QA architecture and therefore needs mentioning.

[4] http://www.w3.org/RDF/

its population size, its mayor, what universities it has, and so on. A semantic QA system can identify the answers to questions which requires the extraction and aggregation of information from more than one document, since it is focused on entities, and not on words.

The entire Wikipedia graph can be represented using these three concepts: nodes, relations and properties. The distinction is made between articles (entities, redirects, disambiguation and lists), categories (classification) and templates (semi-structured format for representing object properties). This graph can be merged with Semantic Web datasets such as dbpedia[5], Yago [5], geonames[6], etc., to give us enough information to address a wide variety of questions.

3 Semantic Question Interpretation

The task of interpreting questions is difficult due to the ambiguity which characterizes natural language. While for humans it is relatively easy to analyse questions and derive an annotation which makes sense, computers, lacking general purpose inference and common-sense, cannot deal effectively with ambiguity. An important part of assigning a semantic interpretation relates to the way data is represented. For example, instead of finding people that are professional football players and who were born in Brazil, a system using Wikipedia does not need to compose the meaning of a span of text by examining individual words, but instead exploits the *Brazilian footballers* category to extract the entities directly. In a nutshell, the difficulty resides in the fact that very similar natural language expressions can have very different semantic interpretations. In order to tackle this problem, the proposed architecture tries to cover as many semantic interpretations as possible by trying to identify distinct semantic constraints. This means that it can exploit the different ways information is represented in Wikipedia, and therefore is more likely to find an answer.

EQUAL is an enhanced version of the WikipediaListQA@wlv system which took part in the GikiP 2008 pilot [4], and builds upon its predecessor by taking steps towards implementing the semantic QA architecture presented in Section 2). For *EQUAL*, interpreting natural language questions means assigning semantics to each unit in a way that would deduce the meaning of an expression, say "Brazilian football players", from the meaning of its individual words. This process is hindered by the lack of a formal semantics dictionary and by the fact that multi-word expressions can have semantics which differ from those of the individual words comprising them. The unit "American football players" would not only mean a different nationality but also a different sport. Wikipedia and its category folksonomy is used to address this problem: in determining the meaning of a natural language expression, we reuse the meaning of its sub-expressions as implied by the Wikipedia Category Graph.

In our model, a **question interpretation** is a set of semantic constraints which can be represented using entities, relations and properties. A question can have several interpretations, each corresponding to a different understanding of the question. There are several types of constraints used by the *EQUAL* system:

- **EAT** (expected answer type) – indicates the type of entities sought: only entities of this type are considered answer candidates;
- **entity** – denotes an entity corresponding to a Wikipedia article;
- **relation** – indicates the relation between entities, usually represented by the verb;
- **property** – restricts a set of entities to a subset which have a certain property (e.g. *population size*)
- **geographic** – restricts the entities to having a particular relation with a geographic entity (e.g. *born in a country*, *contained in a region*);
- **temporal** – constrains the temporal interval of valid entities;
- **introduction** – marks the phrase used to start the question and indicates redundancy.

List || the Italian places || where Ernest Hemingway | visited | during his life.
{$_{intro}$List} the {$_{eat}$Italian places} where {$_{entity}$Ernest Hemingway} {$_{rel}$visited} {$_{temp}$during his life}.

Name || Romanian writers || who were living | in USA | in 2003
{$_{intro}$Name} {$_{eat}$Romanian writers} who were {$_{rel}$living} in USA {$_{temp}$in 2003}

Figure 1: Example of constraints in questions

Figure 1 demonstrates the shallow parsing (see Section 3.1) and relevant constraints applied to two questions. Identifying these constraints is related to the more general problem of parsing. The semantics of the constraints themselves in the context of Wikipedia are defined by *constraint verifiers*, i.e. the actual implementation which verifies whether a particular constraint holds. In this model, a constraint can have several verifiers, which can be specialised for certain subsets of the data or that can use external data sources. Issues related to parsing and verifiers are presented in the following sections.

3.1 Parsing

Parsing is usually employed to map the syntactic parse tree to semantic representations (e.g. subject – actor, verb – action). Standard parsers select the most probable parse(s) given their training data. While this is sufficient for some NLP applications, QA systems need the guarantee that the correct meaning of the question can be deduced from the syntactic trees identified by the parser. Some parsers (e.g. the Stanford Parser[7] [2]) do generate the top n trees, which improves the chances that the correct parse is available. However, there are very few models trained for interrogative sentences, and hardly any which are trained on a corpus of questions similar to those in GikiCLEF.

In order to deal with ambiguity, the semantic QA architecture uses a parallel algorithm to generate all plausible parse trees and to create question interpretations by annotating the syntactic units with their corresponding semantic counterparts. Erroneous interpretations are unlikely to yield any results. A practical system with reasonable coverage would need to employ a ranking scheme to guide the exploration of the parse tree space. The system could then further process each interpretation in parallel, and, based on interaction with the user, decide at run-time how to prune the search space, rather than rely on corpus-determined thresholds.

In *EQUAL*, determining the syntactic units within a question is performed via a shallow parsing mechanism based on certain classes of words (prepositions, relative pronouns, conjunctions, adverbs, verbs) as delimiters. *EQUAL* uses a simplified sequential model of the general architecture (see Section 2), allowing it to stop when it finds a semantic interpretation which yields results. The system is based on a simple rule-driven approach which was developed using the GikiCLEF 2009 training question set.

The first step in parsing is to split the question into two parts: the **domain** of valid candidates (a SELECT clause) and the **filtering** criterion (the constraints, a WHERE clause). This is achieved by removing the introductory phrase (*List...*, *Name...*) and identifying the first verb or pronoun which is considered to be the delimiter. In the second step, the two parts are processed differently. The domain clause is mapped to the most relevant category (the EAT constraint) using as many words as possible. If the match is not straightforward, then the longest match containing the first noun in plural form is identified, and the remaining words are mapped to an additional constraint. The filtering criterion is split into sub-filters using prepositions. The type of the sub-filters (i.e. the semantic constraint) and the way they are linked are dealt with by the semantic interpretation module (see below).

[5] http://wiki.dbpedia.org/About
[6] http://www.geonames.org/ontology/
[7] http://nlp.stanford.edu/software/lex-parser.shtml

3.2 Semantic Interpretation and Constraints

This processing phase assigns semantic interpretations to syntactic units. This means dealing with ambiguity in the context of the semantic constraints that the system 'understands', and the way that the relevant information is present in Wikipedia and any associated data sources. The main ambiguities that are dealt with during this phase are:

- **referential ambiguity**: eponymous entities have the same name and need to be resolved correctly (*"Which American writers were born in Paris?" – Paris* can refer to several entities commonly known as Paris: people, cities, characters, depending on which one is used the retrieved results can be different);
- **structural ambiguity**: the same constraint can be attached to more than one head (*"Which countries have won a futsal European championship played in Spain?"*);
- **type ambiguity**: the constraint can be interpreted in more than one way (*"What novels did Orwell publish before 1984?"* – temporal constraint = year; entity constraint = novel).

The system tries to identify a semantic constraint and to find the correct dependent for each syntactic constituent. Where there is ambiguity, the distinct interpretations are addressed separately. The system examines each syntactic constituent to determine its possible semantic type. For example phrases such as *"greater than"*, *"more than"* could trigger a numeric property filter, while the preposition *"in"* marks both geographical and temporal constraints. This compatibility is not stored beforehand, but is embedded in the actual constraint verifiers used.

4 Answer Extraction

The GikiCLEF question set is composed of entity list questions. While these questions are regarded as more difficult than factoid ones (most factoid questions can be rephrased as list questions which have exactly one answer), they have two important characteristics which make them accessible:

1. encyclopaedic domain — since all questions are asking about entities represented in Wikipedia, certain assumptions regarding the "domain" can be made;
2. structural uniformity — the way the questions are formulated is syntactically uniform, most questions conforming to a limited set of patterns.

4.1 Data Representation

In the GikiCLEF competition, (with very few exceptions) identifying answers requires extracting information from more than one document, using basic notions of structure (tables, infoboxes, layout) and interlinking. As mentioned in Section 2, Wikipedia is viewed as a graph represented using the three concepts: nodes, relations and properties, and the distinction is made between articles (entities, redirects, disambiguation and lists), categories (classification) and templates (semi-structured format for representing object properties).

From the redirect and disambiguation pages, entity aliases are extracted to be used in the question analysis. The articles describing entities are indexed and the links between them are stored in a database, allowing us to query the set of pages that are linked to/from a given article. The categories and their links are stored in a separate table for navigating the category DAG. Apart from the straightforward query for the set of categories of a page X, *EQUAL* can also check if a given page X is directly or indirectly within a category Y, determining, level by level, the pages indirectly connected to category Y, with or without transitivity filtering. Transitivity filtering refers to categories which do not have a plural noun in their title, or which have a corresponding article with exactly the same name (e.g. *London*). This rule is designed to

limit the recursive exploration of the Wikipedia graph. The text of the articles was indexed because some constraint verifiers need to extract information from the text at query time.

The category folksonomy is used to derive type information for entities. Being a folksonomy, the relations between categories are not specified, which results in ambiguities. There are both type and topic categories, but this distinction in not explicit in the mark-up. Although most of the time a direct link from an article to a category denotes an `rdf:type` relation, precautions should be taken when assuming transitivity. More advanced methods can be employed to disambiguate these links (see dbpedia and Yago), but *EQUAL* only uses a simple rule which limits transitivity: subcategories are processed in a DFS order only if no answers have been found and categories with no plural noun in their name are skipped.

The advantage of the categories graph is that it exhibits various granularity levels. For example, it is possible to directly map the question span "Brazilian football players" to the category *Brazilian footballers*, instead of having to deduce the semantics of the individual words, such as checking which "football players" were "born in Brazil" (see Figure 5). These issues are most relevant to the EAT verifier.

4.2 Constraint Verifiers

The verifiers corresponding to the semantic constraints from Section 3 are largely those inherited from the WikipediaListQA@wlv system. The idea behind verifiers is that a generic constraint (e.g. geographic containment) can be realised in Wikipedia by different means (demonym modifiers, e.g. *Nepalese*, categories, e.g. *Mountains of Nepal*, tables, infoboxes, text). By using an adaptor design pattern, *EQUAL* can use several methods for the same semantic constraint. This makes it possible to exploit both information extraction algorithms and external datasets. For example, a geographical reasoning Web-Service can be used to evaluate relations such as containment, neighbouring, distance, and others, while the dbpedia dataset can also be used to extract properties of entities.

The constraint verifiers in *EQUAL* are rather rudimentary and only implement two methods: *create*, which creates a new object via a factory method using a span from the question, and *apply*, to test whether an entity corresponds to the particular semantic constraint. These verifiers are quite general and offer a good balance between accuracy and coverage, but more specialised implementations could perform better. The reason for not adding new verifiers is that a manual approach is not scalable, and improvements could be the result of over-fitting the actual questions, solving the test not the task. Each verifier is discussed below.

Entity constraint. Filtering a set of entities based on a relation with a given entity X is enforced by checking which of the articles mentioned by X are part of the initial set and which of the remaining articles mention X. *EQUAL* uses both the links table and also the article text to identify mentions, to account for cases where the link was not marked by the Wikipedia editors.

Relation constraint. Identifying mentions between pages is a high recall step. *EQUAL* must then confirm that the context of the mention page is 'compatible' with the constraint identified in the question. Initial experiments revealed that in many cases testing the compatibility needs more than synonym expansion or WordNet similarity measures. Therefore this constraint was ignored in the GikiCLEF test set. This means that in questions such as GC09-01 and GC09-09 *EQUAL* answers a slightly different question than the one asked. The effects on performance vary, but usually demonstrate a decrease in precision.

Temporal constraint. In questions that have a time constraint, this is applied to further filter the relation constraint. The implementation uses a temporal annotator developed for the QALL-ME project[8] to transform temporal expressions to time intervals, and then tests if the context of the mention is compatible with the constraint from the question. This approach suffers from the same problems as above. Therefore, for GikiCLEF *EQUAL* used a simpler test: if the constraint is one year, then it must be textually present somewhere in the article's content. All other forms of temporal constraints are ignored, even if in other interpretations the question span could be considered an entity constraint.

Property constraint. *EQUAL* uses both the infobox content and the article text to extract numeric values. The comparatives and other quantifiers from the question are respected. If the noun denoting the

[8] http://qallme.fbk.eu/

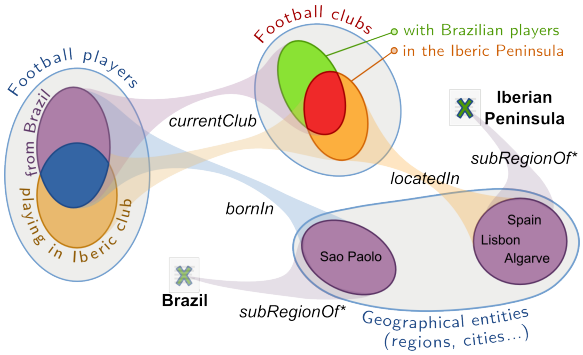

Figure 2: Which Brazilian football players play in clubs in the Iberian Peninsula?

property is not found or a numeric value cannot be extracted, the constraint is not satisfied and the candidate article is discarded.

Geographic constraint. This constraint makes use of a list of demonyms extracted from Wikipedia and a small set of patterns to determine whether a given entity is part of a category such as "ncp of/from/in Country" (e.g. "Mountains of Nepal") or "Demonym ncp" (e.g. "Romanian poets") where ncp stands for a common noun in plural form. This strategy is also used when searching for the best category match for the EAT. If such a pattern does not apply, *EQUAL* interprets this as an entity constraint, as a fall-back.

Introduction constraint. A list of interrogative pronouns (e.g. which), imperative verbs (e.g. name) and declarative constructions (e.g. I'm interested.....) which was manually compiled from the QALL-ME benchmark [1] is used to identify the spans at the beginning of the question. These spans can be considered to mark list questions, but in the context of GikiCLEF this is redundant information and these spans are ignored in all subsequent processing.

5 Discussion

Processing some of the questions requires query-time verifier evaluation for a large number of articles, which is rather inefficient. This is especially true when the EAT category is much more general than the answers sought, because exploring its sub-articles yields a very large set of candidate articles. Questions which have more than one set of entities can also be very slow to process. If the system did not return any answers after 30 minutes, the execution was stopped. This happened for roughly half of the 25 questions which *EQUAL* did not answer. To address this issue, two strategies are currently being investigated: either using an RDF store for faster execution of some patterns, or selecting a better entry point in Wikipedia using spreading activation.

Processing English seems to confer some advantages, due to the larger size of the English Wikipedia. *EQUAL* had 69 correct answers out of the total of 138 answers for 25 of the 50 questions for English, which give a precision of 50% and a score of 34.5. By simply mapping these results in all the other languages using the interwiki link, the cumulative results total 385 correct answers out of 813 for the same questions in all the languages (precision 47%, score 181.93).

While *EQUAL* ranked first in GikiCLEF, the scores are less impressive in terms of recall: the system identified one third of the complete correct answer pool which contains all the correct answers from all the systems. An analysis is under way to investigate why so many answers were missed. This could prove crucial regarding which enhancements should be addressed first.

Related Research – NLIs and ontologies

The methodology employed shares similarities with natural language interfaces (NLIs) to databases and ontologies. The PRECISE system [3] uses a graph–flow algorithm to map tokens from the query to the

lexicon of relations, attributes and values of a database. Given the constraints imposed by the database schema, the validity of a question interpretation can easily be established. They show "how questions that can be mapped to non-recursive Datalog clauses are semantically tractable." EQUAL has a much broader coverage, and the mechanisms are different, however recursion needs to be investigated. Likewise, the formalism behind ontologies (e.g. Description logic, OWL) are not present in the large RDF repositories, since these use general purpose terminologies meant to facilitate data reuse rather than advanced inference.

Amongst the projects that employ information extraction techniques to extract semantic relations from Wikipedia, the most relevant to this work is WikiTaxonomy [6] which extracts explicit and implicit relations by splitting the category title. For example, from *Category:Movies directed by Woody Allen* relations such as *directedBy* and *type(movie)* are extracted. A total of 3.4 million *type* and 3.2 million spatial relations (e.g. *locatedIn*), along with 43,000 *memberOf* relations and 44,000 other relations such as *causedBy* and *writtenBy* were extracted. This is similar to the geographical verifier used in *EQUAL* and to the methods the system employs to identify the best category for the EAT mapping.

6 Conclusions and future work

This paper presented an entity-centric architecture for semantic QA and a prototype system, *EQUAL*. At the core of the architecture is a set of semantic constraints which are used to represent the possible interpretations of questions. When extracting the answers for each interpretation, the constraints are verified using the underlying data and the set of available verifiers, and the candidate entities which satisfy all the constraints are selected. The system performed very well in the GikiCLEF 2009 task, ranking first among 8 systems: it achieved a precision of 47% and a relative recall of 32%, proving adequate for the task. Further improvements are required in order to increase recall and to reduce the runtime complexity of the method.

Possible ways of improving *EQUAL* include adding verifiers focused on more reliable processing of text within articles, automatically learning constraint verifiers using seed data and adding quantifier semantics. In addition, creating a question corpus to investigate training different parsing models or using an RDF store as the underlying model could prove beneficial. In terms of general semantic QA, architecture it is necessary to implement the feedback module which would make *EQUAL* an interactive QA system, and also to devise ranking models for combining the results of several interpretations/verifiers, in order to present the user with the most relevant information.

Acknowledgements

The development of the *EQUAL* system was partly supported by the EU-funded project QALL-ME (FP6 IST-033860).

References

[1] Elena Cabrio, Milen Kouylekov, Bernardo Magnini, Matteo Negri, Laura Hasler, Constantin Orasan, David Tomas, Jose Luis Vicedo, Guenter Neumann, and Corinna Weber. The QALL-ME Benchmark: a Multilingual Resource of Annotated Spoken Requests for Question Answering. In Nicoletta Calzolari, Khalid Choukri, Bente Maegaard, Joseph Mariani, Jan Odjik, Stelios Piperidis, and Daniel Tapias, editors, *Proceedings of the Sixth International Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco, may 2008. European Language Resources Association (ELRA). http://www.lrec-conf.org/proceedings/lrec2008/.

[2] Dan Klein and Christopher D. Manning. Accurate unlexicalized parsing. In *ACL*, pages 423–430, 2003.

[3] Ana-Maria Popescu, Oren Etzioni, and Henry Kautz. Towards a theory of natural language interfaces to databases. In *IUI '03: Proceedings of the 8th international conference on Intelligent user interfaces*, pages 149–157, New York, NY, USA, 2003. ACM.

[4] Diana Santos, Nuno Cardoso, Paula Carvalho, Iustin Dornescu, Sven Hartrumpf, Johannes Leveling, and Yvonne Skalban. Getting geographical answers from Wikipedia: the GikiP pilot at CLEF. In Francesca Borri, Alessandro Nardi, and Carol Peters, editors, *Working Notes for the CLEF 2008 Workshop*, Aarhus, Denmark, September 2008. CLEF 2008 Organizing Committee.

[5] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: A Core of Semantic Knowledge. In *16th international World Wide Web conference (WWW 2007)*, New York, NY, USA, 2007. ACM Press.

[6] Cäcilia Zirn, Vivi Nastase, and Michael Strube. Distinguishing between Instances and Classes in the Wikipedia Taxonomy. In *ESWC*, pages 376–387, 2008.