Deep Answers for Naturally Asked Questions on the Web of Data

Mohamed Yahya, Klaus Berberich, Shady Elbassuoni,
Maya Ramanath[‡], Volker Tresp[†], Gerhard Weikum

Max Planck Institute for Informatics, Germany
myahya,kberberi,elbass,weikum@mpi-inf.mpg.de

[‡]Dept. of CSE, IIT-Delhi, India
ramanath@cse.iitd.ac.in

[†]Siemens AG, Corporate Technology, Munich, Germany
volker.tresp@siemens.com

ABSTRACT

We present DEANNA, a framework for natural language question answering over structured knowledge bases. Given a natural language question, DEANNA translates questions into a structured SPARQL query that can be evaluated over knowledge bases such as Yago, Dbpedia, Freebase, or other Linked Data sources. DEANNA analyzes questions and maps verbal phrases to relations and noun phrases to either individual entities or semantic classes. Importantly, it judiciously generates variables for target entities or classes to express joins between multiple triple patterns. We leverage the semantic type system for entities and use constraints in jointly mapping the constituents of the question to relations, classes, and entities. We demonstrate the capabilities and interface of DEANNA, which allows advanced users to influence the translation process and to see how the different components interact to produce the final result.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Query Formulation; I.2.1 [**Applications and Expert Systems**]: Natural language interfaces

Keywords

question-answering, knowledge base, semantic search, disambiguation, usability

1. INTRODUCTION

Recently, very large, structured, and semantically rich knowledge bases have become available. Examples are Yago [8, 13], Dbpedia [3], and Freebase [5]. Dbpedia forms the nucleus of the Web of Linked Data [4], which interconnects hundreds of RDF data sources with a total of 30 billion subject-property-object (SPO) triples.

Users can search in these knowledge bases using structured query languages like SPARQL, but only expert programmers are able to precisely specify their information needs and cope with the schema-wise highly heterogeneous data. For less initiated users the only option to search this rich data is by keyword search (e.g., via services like sig.ma [14]). However, the most convenient approach to express information needs over knowledge bases and linked data sources is by natural-language questions.

As an example, consider a quiz question like "Which female actor played in Casablanca and is married to a writer who was born in Rome?". One can think of different formulations of the same question, such as "Which actress from Casablanca is married to a writer from Rome?". A possible SPARQL formulation, assuming the user is familiar with the schema of the underlying knowledge base(s), could consist of the following six triple patterns (to be joined by shared-variable bindings): `?x hasGender female`, `?x isa actor`, `?x actedIn Casablanca_(film)`, `?x marriedTo ?w`, `?w isa writer`, `?w bornIn Rome`. This query would yield good results, but it is difficult for the user to come up with the precise choices for relations, classes, and entities. This would require knowledge of the contents of the knowledge base, which no average user is expected to have. Our goal is to automatically create such structured queries by mapping the user's question into this representation. Note that keyword search is usually not a viable alternative when the information need involves joining multiple triples to construct the final result, not withstanding good attempts like [12]. In the example, the obvious keyword query "female actress Casablanca married writer born Rome" lacks a clear specification of the relations that should connect the different entities.

Natural-language question answering, QA for short, has a long history in NLP and IR research. However, the best performing approaches simply map questions into keyword search for finding (passages about) answer candidates; the intelligence of these systems lies in their deeper methods for ranking candidates. The IBM Watson system for Deep QA has demonstrated great advances and impressive performance in the Jeopardy quiz show, but does not consider structured querying at all. Its key strength lies in classifying and decomposing questions and in ranking and type-checking candidate answers [6, 9]. There are few prior attempts to automatically map questions into structured queries (e.g., [7, 10]). However, the scope of these approaches has been very limited. Only recently, interest in general-purpose question-to-query translation has been rekindled in the 2011 QALD workshop [1].

In our approach, we introduce new elements towards mak-

WWW 2012 Companion, April 16–20, 2012, Lyon, France.
ACM 978-1-4503-1230-1/12/04.

ing translation of questions into SPARQL triple patterns more expressive and robust. Most importantly, we harness the rich information in a knowledge base like Yago, which knows not only entities and relations, but also surface names and textual patterns by which web sources refer to them. For example, Yago knows that "Casablanca" can refer to the Morrocan city or the film, and "played in" or simply "from" are patterns that can denote the `actedIn` relation. In addition, we can leverage the rich type system of semantic classes. For example, knowing that Casablanca is a film, for translating "from" we can focus on relations with a type signature whose range includes films.

Based on these ideas, we have developed a framework and prototype system coined DEANNA (DEep Answers for maNy Naturally Asked questions). The framework comprises a full suite of components for question decomposition, mapping constituents into the semantic concept space provided by the Yago knowledge base, generating alternative candidate mappings, and computing a coherent mapping of all constituents into a set of triple patterns that can be directly executed on Yago. Our demo shows both end-to-end question-answering as well as the internal working of the components and their interplay.

2. FRAMEWORK

The main difficulties that DEANNA addresses lie in disambiguating the information needs expressed in the verbal phrases and noun phrases in the question, and doing this in a globally coherent manner. Ambiguity problems occur at different stages:

- mapping names to entities, for example, identifying that "Casablanca" most likely denotes `Casablanca_(film)`, although it could also mean `Casablanca, Morocco`;
- mapping phrases to classes for which the knowledge base has instances, for example, associating "author" with `writer` (assuming `author` is not a known class);
- mapping verbal phrases to relations, for example, inferring that `actedIn` is the best-matching relation for the phrase "played in" – as opposed to say `playedForTeam` (from the sports domain);
- inferring additional query conditions, for example, mapping the adjective "female" into the property-object pattern `hasGender female` which is directly supported by the knowledge base.

DEANNA accepts a natural language question as input and uses a knowledge base including a dictionary of relation patterns. The goal is to produce a structured query which captures the question over the knowledge base. Consider a simplified variant of our example question: "Who played in Casablanca and is married to a writer who was born in Rome?". A structured SPARQL query would look as follows:

```
SELECT ?x WHERE{
  ?x type person . ?x actedIn Casablanca_(film) .
  ?x marriedTo ?w .
  ?w isa writer . ?w bornIn Rome }
```

3. PROTOTYPE SYSTEM

Figure 1 shows the architecture of DEANNA, which is composed of six main components:

1. **Phrase detection**. Phrases are detected that potentially correspond to *semantic items* (relations, entities and classes) such as 'Who', 'played in', 'movie', 'Casablanca' and 'writer'.

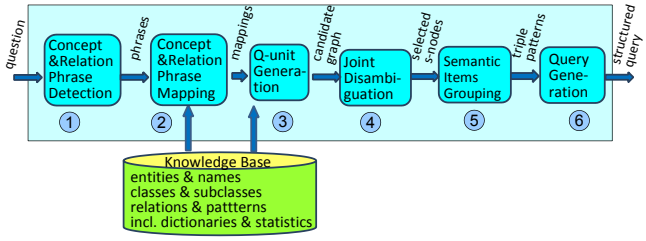

Figure 1: System architecture

2. **Phrase mapping** to semantic items. This includes finding that the phrase 'played in' can either refer to the semantic relation `actedIn` or to `playedForTeam` and the phrase 'Casablanca' can potentially refer to `Casablanca_(film)` or `Casablanca, Morocco`.
3. **Q-unit generation**. Intuitively, a *q-unit* is a triple composed of phrases. Their generation and role will be discussed in detail.
4. **Joint disambiguation**, where the ambiguities in the phrase-to-semantic item mapping are resolved. Here, we determine for our running example that 'played in' refers to the semantic relation `actedIn` and not to `playedForTeam` and the phrase 'Casablanca' refers to `Casablanca_(film)` and not `Casablanca, Morocco`.
5. **Semantic item grouping** to form semantic triples. For example, we determine that the relation `marriedTo` connects the `person` referred to by 'Who' and the `writer` to form the semantic triple `person marriedTo writer`.
6. **Query generation**. If our target language is SPARQL, then the semantic triple `person marriedTo writer` has to be mapped to the triple patterns `?x type person`, `?x marriedTo ?w`, and `?w type writer`.

Phrase detection uses multiple detectors, each of which can detect a certain class of phrases, corresponding to semantic classes, entities, relations and interrogative pronouns. The detectors utilize different named entity recognition and relation detection techniques. Each detector works independently of the others, causing phrases to overlap, resulting in more ambiguity.

For *concepts* (entities and classes), the mapping to semantic items relies on the knowledge base having a dictionary of surface forms to concepts, which looks as follows:

{'Rome','eternal city','Roma'}	→	`Rome`
{'Casablanca'}	→	`Casablanca_(film)`
{'Casablanca', 'White House'}	→	`White_House`

To map relational phrases to semantic relations, a similar dictionary of textual pattern mappings to relations is maintained. The dictionary looks as follows:

{'play','star in','act','leading role'}	→	`actedIn`
{'married', 'spouse','wife'}	→	`marriedTo`

This step is performed by the second stage of our architecture shown in Figure 1. For both concepts and relations, this mapping process will generally generate multiple semantic item candidates per phrase.

Next, *triploids* are detected in the natural language question using a dependency parser [11]. A triploid consists of three tokens potentially corresponding to a relation and its

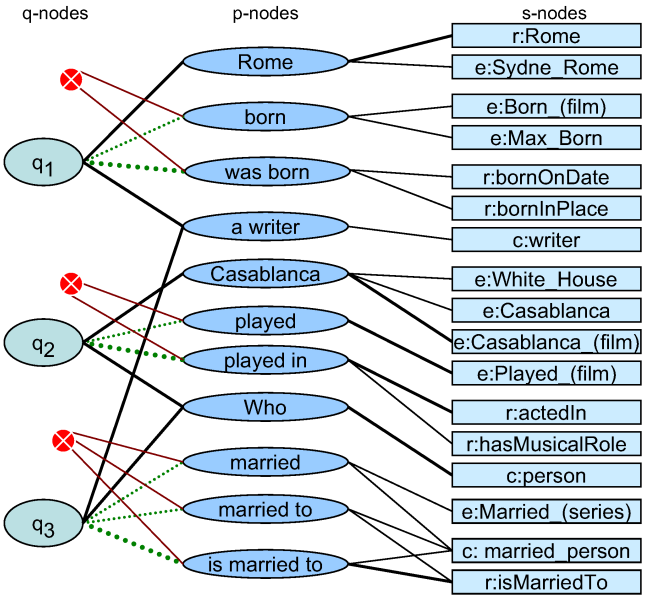

Figure 2: Disambiguation graph

two arguments. Generally speaking, each token of a triploid is the head of a noun phrase or verbal phrase.

Combining triploids and candidate phrases, the framework generates *q-units*, where each q-unit is a phrase-level triple, possibly with multiple candidates for a relation and arguments.

At this point, the framework performs *disambiguation* to resolve the mappings of phrases to semantic items. Because the result of each disambiguation can influence the result of others, we perform disambiguation is a joint manner. We do this by constructing a weighted disambiguation graph and then finding a dense subgraph.

The *disambiguation graph*, such as the one in Figure 2 can be seen as a tripartite graph with a set of semantic item nodes (s-nodes), a set of phrase nodes (p-nodes) and a set of q-unit nodes (q-nodes). S-nodes are connected to each other via weighted *semantic coherence* edges, which indicate how compatible a pair of semantic items is (these are not shown in Figure 2). A p-node is connected to an s-node via a weighted similarity edge that captures the strength of the mapping of the phrase to the semantic item using a weighted combination of a prior and syntactic similarity. Q-nodes collect triples of phrases together. Each of the subject, predicate and object fields in a q-unit is a set of possible phrases, with each member of a q-unit's relation set shown using a dotted edge in Figure 2.

The problem of disambiguation is now reduced to finding a subgraph of the disambiguation graph described above. The desired subgraph is one with maximum density that respects a set of judiciously crafted constraints. The density measure takes into account the weights of similarity and coherence edges. Examples of our constraints include:

- each token can be part of at most one phrase (shown as $\otimes$ nodes in Figure 2);
- each p-node is mapped to one s-node at most;
- each triple should contain a subject, object and predicate;
- each triple should respect the semantic type constraints;

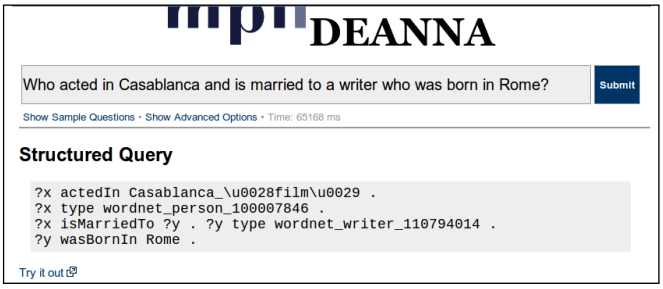

Figure 3: Basic interface

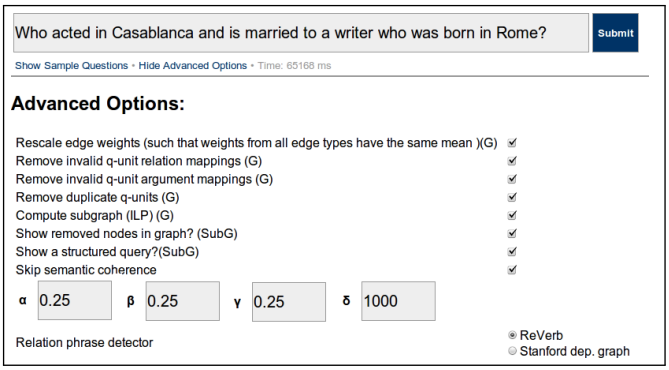

Figure 4: Advanced options

- each triple should contain a semantic class that will map to a type-constrained variable.

We encode our objective and constraints into a integer linear program (ILP), which we then pass to the Gurobi ILP solver [2] which efficiently computes the desired subgraph in which phrase boundaries and mappings are disambiguated.

Once phrases have been unambiguously mapped to semantic items, DEANNA forms semantic triples induced by q-units and type constraints between the different semantic items. The final remaining step is to construct a structured query from the semantic triples. DEANNA constructs SPARQL queries from semantic triples by simply mappings semantic classes to type-constrained variables. Variable assignment is also guided by q-units, as multiple occurrences of the same semantic class might need to be mapped to different variables.

4. DEMO

DEANNA is an interactive question answering system. It allows users to input questions in natural language and generates queries over a knowledge base that return answers. Users can look into the intermediate steps of the translation process and influence the process through options provided by the interface. DEANNA currently runs over the Yago knowledge base (which contains 10 million entities, 140 relations, and 450 million facts). A live version of the demo can be accessed by going to `http://www.mpi-inf.mpg.de/yago-naga/deanna/`.

DEANNA operates in either default or advanced mode. The default mode targets the average user, where the system uses a default configuration which we have found to be robust. Figure 3 shows a screenshot of the system in default mode, with a question and the resulting structured query.

The advanced mode targets expert users. It allows them

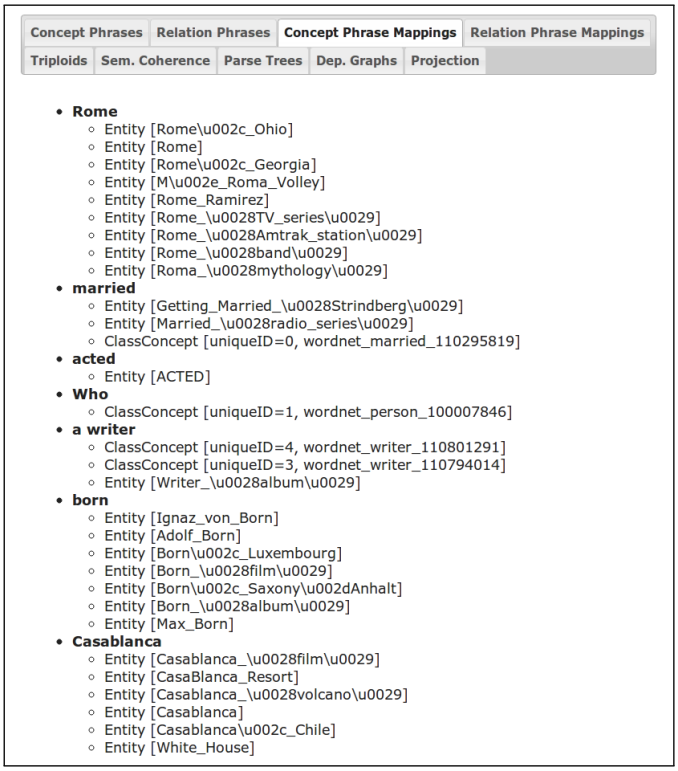

Figure 5: Intermediate output - concept phrase mappings

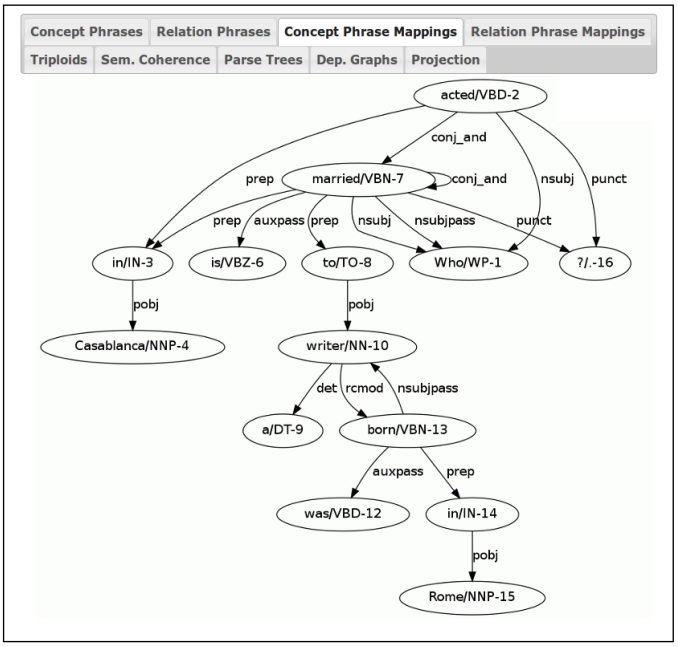

Figure 6: Intermediate output - dependency graphs

to manipulate how different components of the system influenced the results. The system provides explanations for each option, these include options for how node weights are computed in the disambiguation graph and how far the translation process should go, and the type of relational phrase recognizer to use. A screenshot of the system in advanced mode is shown in Figure 4. We have found the advanced mode to be very helpful in developing our QA framework.

The 'Download Graph' icon (not shown) gives the user a PDF version of the disambiguation graph up to the point to which the translation process has been run (completely in default mode and depending on the user's options in advanced mode).

The intermediate output a user can see include relation and concept phrases and their candidate mappings, parse trees and dependency graphs, triploids and semantic coherence between the various semantic items. Each type of intermediate output is shown by clicking on its tab. Figure 5 shows a screenshot of the concept phrase mappings tab for our running example. It shows each candidate concept phrases and its possible semantic item mappings. Figure 6 shows a screenshot of the dependency graphs used in the translation process.

5. REFERENCES

[1] 1st Workshop on Question Answering over Linked Data (QALD-1). `http://www.sc.cit-ec.uni-bielefeld.de/qald-1`, May 2011.

[2] Gurobi Optimizer. `http://www.gurobi.com/`, 2011.

[3] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. G. Ives. DBpedia: A Nucleus for a Web of Open Data. *ISWC/ASWC*, 2007.

[4] C. Bizer, T. Heath, T. Berners-Lee, and M. Hausenblas. 4th Linked Data on the Web Workshop (LDOW2011). *WWW*, 2011.

[5] K. D. Bollacker, C. Evans, P. Paritosh, T. Sturge, J. Taylor. Freebase: a Collaboratively Created Graph Database for Structuring Human Knowledge. *SIGMOD*, 2008.

[6] D. A. Ferrucci et al. Building Watson: An Overview of the DeepQA Project. *AI Magazine*, 31(3), 2010.

[7] A. Frank, H.-U. Krieger, F. Xu, H. Uszkoreit, B. Crysmann, B. Jörg, U. Schäfer. Question Answering from Structured Knowledge Sources. *J. App. Logic*, 5(1), 2007.

[8] J. Hoffart, F. M. Suchanek, K. Berberich, E. Lewis-Kelham, G. de Melo, G. Weikum. YAGO2: Exploring and Querying World Knowledge in Time, Space, Context, and many Languages. *WWW*, 2011.

[9] A. Kalyanpur, J. W. Murdock, J. Fan, C. A. Welty. Leveraging Community-Built Knowledge for Type Coercion in Question Answering. *ISWC*, 2011.

[10] Y. Li, H. Yang, H. V. Jagadish. NaLIX: A Generic Natural Language Search Environment for XML Data. *ACM Trans. Database Syst.*, 32(4), 2007.

[11] M.-C. D. Marneffe, B. Maccartney, C. D. Manning. Generating Typed Dependency Parses from Phrase Structure Parses. *LREC*, 2006.

[12] J. Pound, I. F. Ilyas, G. E. Weddell. Expressive and Flexible Access to Web-extracted Data: a Keyword-based Structured Query Language. *SIGMOD*, 2010.

[13] F. M. Suchanek, G. Kasneci, G. Weikum. Yago: A Core of Semantic Knowledge. *WWW*, 2007.

[14] G. Tummarello, R. Cyganiak, M. Catasta, S. Danielczyk, R. Delbru, S. Decker. Sig.ma: Live Views on the Web of Data. *J. Web Sem.*, 8(4), 2010.