Natural Language Questions for the Web of Data

Mohamed Yahya[1], Klaus Berberich[1], Shady Elbassuoni[2]
Maya Ramanath[3], Volker Tresp[4], Gerhard Weikum[1]
[1] Max Planck Institute for Informatics, Germany
[2] Qatar Computing Research Institute
[3] Dept. of CSE, IIT-Delhi, India [4] Siemens AG, Corporate Technology, Munich, Germany
{myahya,kberberi,weikum}@mpi-inf.mpg.de
selbassuoni@qf.org.qa
ramanath@cse.iitd.ac.in volker.tresp@siemens.com

Abstract

The Linked Data initiative comprises structured databases in the Semantic-Web data model RDF. Exploring this heterogeneous data by structured query languages is tedious and error-prone even for skilled users. To ease the task, this paper presents a methodology for translating natural language questions into structured SPARQL queries over linked-data sources.

Our method is based on an integer linear program to solve several disambiguation tasks jointly: the segmentation of questions into phrases; the mapping of phrases to semantic entities, classes, and relations; and the construction of SPARQL triple patterns. Our solution harnesses the rich type system provided by knowledge bases in the web of linked data, to constrain our semantic-coherence objective function. We present experiments on both the question translation and the resulting query answering.

1 Introduction

1.1 Motivation

Recently, very large, structured, and semantically rich knowledge bases have become available. Examples are Yago (Suchanek et al., 2007), DBpedia (Auer et al., 2007), and Freebase (Bollacker et al., 2008). DBpedia forms the nucleus of the Web of Linked Data (Heath and Bizer, 2011), which interconnects hundreds of RDF data sources with a total of 30 billion subject-property-object (SPO) triples.

The diversity of linked-data sources and their high heterogeneity make it difficult for humans to search and discover relevant information. As linked data is in RDF format, the standard approach would be to run structured queries in triple-pattern-based languages like SPARQL, but only expert programmers are able to precisely specify their information needs and cope with the high heterogeneity of the data (and absence or very high complexity of schema information). For less initiated users the only option to query this rich data is by keyword search (e.g., via services like sig.ma (Tummarello et al., 2010)). None of these approaches is satisfactory. Instead, the by far most convenient approach would be to search in knowledge bases and the Web of linked data by means of 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?". The answer could be found by querying several linked data sources together, like the IMDB-style LinkedMDB movie database and the DBpedia knowledge base, exploiting that there are entity-level `sameAs` links between these collections. One can think of different formulations of the example question, such as "Which actress from Casablanca is married to a writer from Rome?". A possible SPARQL formulation, assuming a user familiar with the schema of the underlying knowledge base(s), could consist of the following six triple patterns (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 complex query, which involves multiple joins, would yield good results, but it is difficult for the user to come

Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pages 379–390, Jeju Island, Korea, 12–14 July 2012.

up with the precise choices for relations, classes, and entities. This would require familiarity with 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. Keyword search is usually not a viable alternative when the information need involves joining multiple triples to construct the final result, notwithstanding good attempts like that of Pound et al. (2010). In the example, the obvious keyword query "female actress Casablanca married writer born Rome" lacks a clear specification of the relations among the different entities.

1.2 Problem

Given a natural language question q_{NL} and a knowledge base KB, our goal is to translate q_{NL} into a formal query q_{FL} that captures the information need expressed by q_{NL}.

We focus on input questions that put the emphasis on entities, classes, and relations between them. We do not consider aggregations (counting, max/min, etc.) and negations. As a result, we generate structured queries of the form known as conjunctive queries or select-project-join queries in database terminology. Our target language is SPARQL 1.0, where the above focus leads to queries that consist of multiple triple patterns, that is, conjunctions of SPO search conditions. We do not use any pre-existing query templates, but generate queries from scratch as they involve a variable number of joins with a-priori unknown join structure.

A major challenge is in the ambiguity of the phrases occurring in a natural-language question. Phrases can denote entities (e.g., the city of Casablanca or the movie Casablanca), classes (e.g., actresses, movies, married people), or relations/properties (e.g., `marriedTo` between people, `played` between people and movies). A priori, we do not know if a phrase should be mapped to an entity, a class, or a relation. In fact, some phrases may denote any of these three kinds of targets. For example, a phrase like "wrote score for" in a question about film music composers, could map to the composer-film relation `wroteSoundtrackForFilm`, to the class of `movieSoundtracks` (a subclass of music pieces), or to an entity like the movie "The Score". Depending on the choice, we may arrive at a structurally good query (with triple patterns that can actually be joined) or at a meaningless and non-executable query (with disconnected triple patterns). This generalized disambiguation problem is much more challenging than the more focused task of named entity disambiguation (NED). It is also different from general word sense disambiguation (WSD), which focuses on the meaning of individual words (e.g., mapping them to WordNet synsets).

1.3 Contribution

In our approach, we introduce new elements towards making translation of questions into SPARQL triple patterns more expressive and robust. Most importantly, we solve the disambiguation and mapping tasks jointly, by encoding them into a comprehensive integer linear program (ILP): the segmentation of questions into meaningful phrases, the mapping of phrases to semantic entities, classes, and relations, and the construction of SPARQL triple patterns. The ILP harnesses the richness of large knowledge bases like Yago2 (Hoffart et al., 2011b), which has information not only about entities and relations, but also about surface names and textual patterns by which web sources refer to them. For example, Yago2 knows that "Casablanca" can refer to the city or the film, and "played in" is a pattern 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 "played in" we can focus on relations with a type signature whose range includes films, as opposed to sports teams, for example. Such information is encoded in judiciously designed constraints for the ILP. Although we intensively harness Yago2, our approach does not depend on a specific choice of knowledge base or language resource for type information and phrase/name dictionaries. Other knowledge bases such as DBpedia can be easily plugged in.

Based on these ideas, we have developed a framework and system, called DEANNA (DEep Answers for maNy Naturally Asked questions), that comprises a full suite of components for question decomposition, mapping constituents into the semantic concept space, generating alternative candidate mappings, and computing a coherent mapping of all constituents into a set of SPARQL triple patterns that

can be directly executed on one or more linked data sources.

2 Background

We use the Yago2 knowledge base, with its rich type system, as a semantic backbone. Yago2 is composed of instances of binary relations derived from Wikipedia and WordNet. The instances, called *facts*, provide both ontological information and instance data. Figure 1 shows sample facts from Yago2. Each fact is composed of *semantic items* that can be divided into relations, entities, and classes. Entities and classes together are referred to as *concepts*.

Subject	Predicate	Object
film	subclassOf	production
Casablanca_(film)	type	film
"Casablanca"	means	Casablanca_(film)
"Casablanca"	means	Casablanca,_Morocco
Ingrid_Bergman	actedIn	Casablanca_(film)

Figure 1: Sample knowledge base

Examples of relations are `type`, `subclassOf`, and `actedIn`. Each relation has a type signature: classes for the relation's domain and range. Classes, such as `person` and `film` group entities. Entities are represented in canonical form such as `Ingrid_Bergman` and `Casablanca_(film)`. A special type of entities are literals, such as strings, numbers, and dates.

3 Framework

Given a natural language question, Figure 2 shows the tasks DEANNA performs to translate a question into a structured query. The first three steps prepare the input for constructing a disambiguation graph for mapping the phrases in a question onto entities, classes, and relations, in a coherent manner. The fourth step formulates this generalized disambiguation problem as an ILP with complex constraints and computes the best solution using an ILP solver. Finally, the fifth and sixth step together use the disambiguated mapping to construct an executable SPARQL query.

A question sentence is a sequence of tokens, $q_{NL} = (t_0, t_1, ..., t_n)$. A phrase is a contiguous subsequence of tokens $(t_i, t_{i+1}, ..., t_{i+l}) \subseteq q_{NL}, 0 \leq i, 0 \leq l \leq n$. The input question is fed into the following pipeline of six steps:

1. **Phrase detection**. Phrases are detected that potentially correspond to semantic items such as 'Who', 'played in', 'movie' and 'Casablanca'.
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 that the phrase 'Casablanca' can potentially refer to `Casablanca_(film)` or `Casablanca,_Morocco`. This step merely constructs a candidate space for the mapping. The actual disambiguation is addressed by step 4, discussed below.
3. **Q-unit generation**. Intuitively, a *q-unit* is a triple composed of phrases. Their generation and role will be discussed in detail in the next section.
4. **Joint disambiguation**, where the ambiguities in the phrase-to-semantic-item mapping are resolved. This entails resolving the ambiguity in phrase borders, and above all, choosing the best fitting candidates from the semantic space of entities, classes, and relations. 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 items grouping** to form semantic triples. For example, we determine that the relation `marriedTo` connects `person` referred to by 'Who' and `writer` to form the semantic triple `person marriedTo writer`. This is done via q-units.
6. **Query generation**. For SPARQL queries, semantic triples such as `person marriedTo writer` have to be mapped to suitable triple patterns with appropriate join conditions expressed through common variables: `?x type person`, `?x marriedTo ?w`, and `?w type writer` for the example.

3.1 Phrase Detection

A *detected phrase* p is a pair $< Toks, l >$ where $Toks$ is a phrase and l is a label, $l \in \{concept, relation\}$, indicating whether a phrase is a relation phrase or a concept phrase. P_r is the set of all detected relation phrases and P_c is the set of all detected concept phrases.

One special type of detected relation phrase is the *null phrase*, where no relation is explicitly mentioned, but can be induced. The most prominent example of this is the case of adjectives, such as 'Australian movie', where we know there is a relation being expressed between 'Australia' and 'movie'.

We use multiple detectors for detecting phrases of

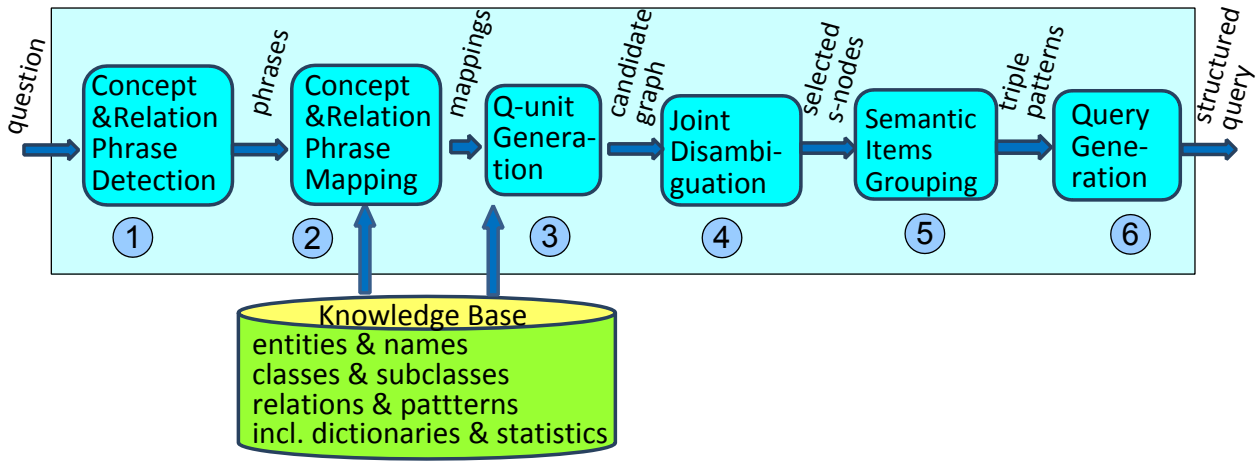

Figure 2: Architecture of DEANNA.

different types. For concept detection, we use a detector that works against a phrase-concept dictionary which looks as follows:

{'Rome','eternal city'} → `Rome`
{'Casablanca'} → `Casablanca_(film)`

We experimented with using third-party named entity recognizers but the results were not satisfactory. This dictionary was mostly constructed as part of the knowledge base, independently of the question-to-query translation task in the form of instances of the `means` relation in Yago2, an example of which is shown in Figure 1

For relation detection, we experimented with various approaches. We mainly rely on a relation detector based on ReVerb (Fader et al., 2011) with additional POS tag patterns, in addition to our own which looks for patterns in dependency parses.

3.2 Phrase Mapping

After phrases are detected, each phrase is mapped to a set of semantic items. The mapping of concept phrases also relies on the phrase-concept dictionary.

To map relation phrases, we rely on a corpus of textual patterns to relation mappings of the form:

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

Distinct phrase occurrences will map to different semantic item instances. We discuss why this is important when we discuss the construction of the disambiguation graph and variable assignment in the structured query.

3.3 Dependency Parsing & Q-Unit Generation

Dependency parsing identifies triples of tokens, or *triploids*, $\langle t_{rel}, t_{arg1}, t_{arg2} \rangle$, where $t_{rel}, t_{arg1}, t_{arg2} \in q_{NL}$ are seeds for phrases, with the triploid acting as a seed for a potential SPARQL triple pattern. Here, t_{rel} is the seed for the relation phrase, while t_{arg1} and t_{arg2} are seeds for the two arguments. At this point, there is no attempt to assign subject/object roles to the arguments.

Triploids are collected by looking for specific dependency patterns in dependency graphs (de Marneffe et al., 2006). The most prominent pattern we look for is a verb and its arguments. Other patterns include adjectives and their arguments, prepositionally modified tokens and objects of prepositions.

By combining triploids with detected phrases, we obtain *q-units*. A q-unit is a triple of sets of phrases, $\langle \{p_{rel} \in P_r\}, \{p_{arg1} \in P_c\}, \{p_{arg2} \in P_c\} \rangle$, where $t_{rel} \in p_{rel}$ and similarly for arg_1 and arg_2. Conceptually, one can view a q-unit as a placeholder node with three sets of edges, each connecting the same q-node to a phrase that corresponds to a relation or concept phrase in the same q-unit. This notion of nodes and edges will be made more concrete when we present our disambiguation graph construction.

3.4 Disambiguation of Phrase Mappings

The core contribution of this paper is a framework for disambiguating phrases into semantic items – covering relations, classes, and entities in a unified manner. This can be seen as a joint task combining

named entity disambiguation for entities, word sense disambiguation for classes (common nouns), and relation extraction. The next section presents the disambiguation framework in detail.

3.5 Query Generation

Once phrases are mapped to unique semantic items, we proceed to generate queries in two steps. First, semantic items are grouped into triples. This is done using the triploids generated earlier. The power of using a knowledge base is that we have a rich type system that allows us to tell if two semantic items are compatible or not. Each relation has a type signature and we check whether the candidate items are compatible with the signature.

We did not assign subject/object roles in triploids and q-units because a natural language relation phrase might express the inverse of a semantic relation, e.g., the natural language expression 'directed by' and the relation `isDirectorOf` with respect to the movies domain are inverses of each other. Therefore, we check which assignment of $arg1$ and $arg2$ is compatible with the semantic relation. If both arrangements are compatible, then we give preference to the assignment given by the dependency parsers.

Once semantic items are grouped into triples, it is an easy task to expand them to SPARQL triple patterns. This is done by replacing each semantic class with a distinct type-constrained variable. Note that this is the reason why each distinct phrase maps to a distinct instance of a semantic class, to ensure correct variable assignment. This becomes clear when we consider the question "Which singer is married to a singer?", which requires two distinct variables each constrained to bind to an entity of type `singer`.

4 Joint Disambiguation

The goal of the disambiguation step is to compute a partial mapping of phrases onto semantic items, such that each phrase is assigned to at most one semantic item. This step also resolves the phrase-boundary ambiguity, by enforcing that only non-overlapping phrases are mapped. As the result of disambiguating one phrase can influence the mapping of other phrases, we consider all phrases jointly in one big disambiguation task.

In the following, we construct a disambiguation graph that encodes all possible mappings. We impose a variety of complex constraints (mutual exclusion among overlapping phrases, type constraints among the selected semantic items, etc.), and define an objective function that aims to maximize the joint quality of the mapping. The graph construction itself may resemble similar models used in NED (e.g., (Milne and Witten, 2008; Kulkarni et al., 2009; Hoffart et al., 2011a)). Recall, however, that our task is more complex because we jointly consider entities, classes, and relations in the candidate space of possible mappings. Because of this complication and to capture our complex constraints, we do not employ graph algorithms, but model the general disambiguation problem as an ILP.

4.1 Disambiguation Graph

Joint disambiguation takes place over a disambiguation graph $DG = (V, E)$, where $V = V_s \cup V_p \cup V_q$ and $E = E_{sim} \cup E_{coh} \cup E_q$, where:

- V_s is the set of semantic items, $v_s \in V_s$ is an *s-node*.
- V_p is the set of phrases, $v_p \in V_p$ is called a *p-node*. We denote the set of p-nodes corresponding to relation phrases by V_{r_p} and the set of p-nodes corresponding to concept phrases by V_{r_c}.
- V_q is a set of placeholder nodes for q–units, called *q-nodes*. They represent phrase triples.
- $E_{sim} \subseteq V_p \times V_s$ is a set of weighted *similarity edges* that capture the strength of the mapping of a phrase to a semantic item.
- $E_{coh} \subseteq V_s \times V_s$ is a set of weighted *coherence edges* that capture the semantic coherence between two semantic items. Semantic coherence is discussed in more detail later in this section.
- $E_q \subseteq V_q \times V_p \times d$, where $d \in \{rel, arg_1, arg_2\}$ is a *q-edge*. Each such edge connects a placeholder q-node to a p-node with a specific role as a relation, or one of the two arguments. A q-unit, as presented earlier, can be seen as a q-node along with its outgoing q-edges.

Figure 3 shows the disambiguation graph for our running example (excluding coherence edges between s-nodes).

4.2 Edge Weights

We next describe how the weights on similarity edges and semantic coherence edges are defined.

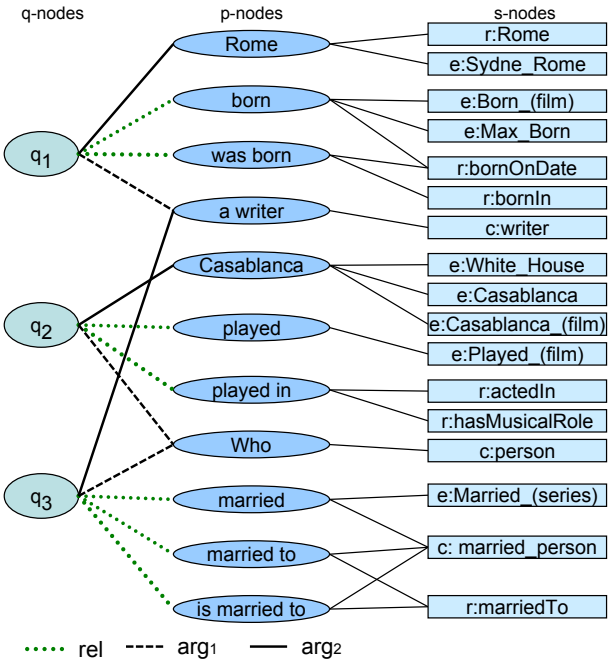

Figure 3: Disambiguation graph for the running example.

4.2.1 Semantic Coherence

Semantic coherence, Coh_{sem}, captures to what extent two semantic items occur in the same context. This is different from semantic similarity (Sim_{sem}), which is usually evaluated using the distance between nodes in a taxonomy (Resnik, 1995). While we expect Sim_{sem}(`George_Bush`, `Woody_Allen`) to be higher than Sim_{sem}(`Woody_Allen`, `Terminator`) we would like Coh_{sem}(`Woody_Allen`, `Terminator`), both of which are from the entertainment domain, to be higher than Coh_{sem}(`George_Bush`, `Woody_Allen`).

For Yago2, we characterize an entity e by its *inlinks* $InLinks(e)$: the set of Yago2 entities whose corresponding Wikipedia pages link to the entity.

To be able to compare semantic items of different semantic types (entities, relations, and classes), we need to extend this to classes and relations. For class c with entities e, its inlinks are defined as follows:

$$InLinks(c) = \bigcup_{e \in c} Inlinks(e)$$

For relations, we only consider those that map entities to entities (e.g. `actedIn`, `produced`), for which we define the set of inlinks as follows:

$$InLinks(r) = \bigcup_{(e_1,e_2) \in r} (InLinks(e_1) \cap InLinks(e_2))$$

The intuition behind this is that when the two arguments of an instance of the relation co-occur, then the relation is being expressed.

We define the semantic coherence (Coh_{sem}) between two semantic items s_1 and s_2 as the Jaccard coefficient of their sets of inlinks.

4.2.2 Similarity Weights

Similarity weights are computed differently for entities, classes, and relations. For entities, we use a normalized prior score based on how often a phrase refers to a certain entity in Wikipedia. For classes, we use a normalized prior that reflects the number of members in a class. Finally, for relations, similarity reflects the maximum n-gram similarity between the phrase and any of the relation's surface forms. We use Lucene for indexing and searching the relation surface forms.

4.3 Disambiguation Graph Processing

The result of disambiguation is a subgraph of the disambiguation graph, yielding the most coherent mappings. We employ an ILP to this end. Before describing our ILP, we state some necessary definitions:

- Triple dimensions: $d \in \{rel, arg_1, arg_2\}$
- Tokens: $T = \{t_0, t_1, ..., t_n\}$.
- Phrases: $P = \{p_0, p_1, ..., p_k\}$.
- Semantic items: $S = \{s_0, s_1, ..., s_l\}$.
- Token occurrences: $\mathcal{P}(t) = \{p \in P \mid t \in p\}$.
- $X_i \in \{0, 1\}$ indicates if p-node i is selected.
- $Y_{ij} \in \{0, 1\}$ indicates if p-node i maps to s-node j.
- $Z_{kl} \in \{0, 1\}$ indicates if s-nodes k, l are *both* selected so that their coherence edge matters.
- $Q_{mnd} \in \{0, 1\}$ indicates if the q-edge between q-node m and p-node n for d is selected.
- C_j, E_j and R_j are $\{0, 1\}$ constants indicating if s-node j is a class, entity, or relation, resp.
- w_{ij} is the weight for a p–s similarity edge.
- v_{kl} is the weight for an s–s semantic coherence edge.
- $t_{rc} \in \{0, 1\}$ indicates if the relation s-node r is type-compatible with the concept s-node c.

Given the above definitions, our objective function is

$$maximize \quad \alpha \sum_{i,j} w_{ij} Y_{ij} + \beta \sum_{k,l} v_{kl} Z_{kl} + \gamma \sum_{m,n,d} Q_{mnd}$$

subject to the following constraints:

1. A p-node can be assigned to one s-node *at most*:

$$\sum_j Y_{ij} \leq 1,\ \forall i$$

2. If a p-s similarity edge is chosen, then the respective p-node must be chosen:

$$Y_{ij} \leq X_i,\ \forall j$$

3. If s-nodes k and l are chosen ($Z_{kl} = 1$), then there are p-nodes mapping to each of the s-nodes k and l ($Y_{ik} = 1$ for some i *and* $Y_{jl} = 1$ for some j):

$$Z_{kl} \leq \sum_i Y_{ik} \text{ and } Z_{kl} \leq \sum_j Y_{jl}$$

4. No token can appear as part of two phrases:

$$\sum_{i \in \mathcal{P}(t)} X_i \leq 1, \forall t \in T$$

5. At most one q-edge is selected for a dimension:

$$\sum_n Q_{mnd} \leq 1,\ \forall m, d$$

6. If the q-edge mnd is chosen ($Q_{mnd} = 1$) then p-node n must be selected:

$$Q_{mnd} \leq X_n,\ \forall m, d$$

7. Each semantic triple should include a relation:

$$E_r \geq Q_{mn'd} + X_{n'} + Y_{n'r} - 2\ \ \forall m, n', r, d = rel$$

8. Each triple should have at least one class:

$$C_{c_1} + C_{c_2} \geq\ Q_{mn''d_1} + X_{n''} + Y_{n'''c_1} + Q_{mn'''d_2} + X_{n'''} + Y_{n'''c_2} - 5,$$
$$\forall m, n'', n''', r, c_1, c_2, d_1 = arg1, d_2 = arg2$$

This is not invoked for existential questions that return Boolean answers and are translated to `ASK` queries in SPARQL. An example is the question "Did Tom Cruise act in Top Gun?", which can be translated to `ASK {Tom_Cruise actedIn Top_Gun}`.

9. Type constraints are respected (through q-edges):

$$t_{rc_1} + t_{rc_2} \geq\ Q_{mn'd_1} + X_{n'} + Y_{n'r} + Q_{mn''d_2} + X_{n''} + Y_{n'''c_1} + Q_{mn'''d_3} + X_{n'''} + Y_{n'''c_2} - 7$$
$$\forall m, n', n'', n''', r, c_1, c_2,$$
$$d_1 = rel, d_2 = arg1, d3 = arg2$$

The above is a sophisticated ILP, and most likely NP-hard. However, even with ten thousands of variables it is within the regime of modern ILP solvers. In our experiments, we used Gurobi (Gur, 2011), and achieved run-times – typically of a few seconds.

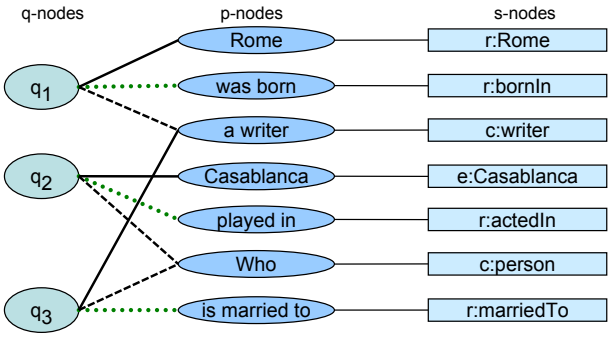

Figure 4: Computed subgraph for the running example.

Figure 4 shows the resulting subgraph for the disambiguation graph of Figure 3. Note how common p-nodes between q-units capture joins.

5 Evaluation

5.1 Datasets

Our experiments are based on two collections of questions: the QALD-1 task for question answering over linked data (QAL, 2011) and a collection of questions used in (Elbassuoni et al., 2011; Elbassuoni et al., 2009) in the context of the NAGA project, for informative ranking of SPARQL query answers (Elbassuoni et al. (2009) evaluated the SPARQL queries, but the underlying questions are formulated in natural language.) The NAGA collection is based on linking data from IMDB with the Yago2 knowledge base. This is an interesting linked-data case: IMDB provides data about movies, actors, directors, and movie plots (in the form of descriptive keywords and phrases); Yago2 adds semantic types and relational facts for the participating entities. Yago2 provides nearly 3 million concepts and 100 relations, of which 41 lie within the scope of our framework.

Typical example questions for these two collections are: "Which software has been published by Mean Hamster Software?" for QALD-1, and "Which director has won the Academy Award for Best director and is married to an actress that has won the Academy Award for Best Actress?" for NAGA. For both collections, some questions are out-of-scope for our setting, because they mention entities or relations that are not available in the underlying datasets, contain date or time comparisons, or involve aggregation such as counting. After re-

moving these questions, our test set consists of 27 QALD-1 training questions out of a total of 50 and 44 NAGA questions, out of a total of 87. We used the 19 questions from the QALD-1 test set that are within the scope of our method for tuning the hyper-parameters (α, β, γ) in the ILP objective function.

5.2 Evaluation Metrics

We evaluated the output of DEANNA at three stages in the processing pipeline: a) after the disambiguation of phrases, b) after the generation of the SPARQL query, and c) after obtaining answers from the underlying linked-data sources. This way, we could obtain insights into our building blocks, in addition to assessing the end-to-end performance. In particular, we could assess the goodness of the question-to-query translation *independently* of the actual answer quality which may depend on particularities of the underlying datasets (e.g., slight mismatches between query terminology and the names in the data.)

At each of the three stages, the output was shown to two human assessors who judged whether an output item was good or not. If the two were in disagreement, then a third person resolved the judgment.

For the *disambiguation* stage, the judges looked at each q-node/s-node pair, in the context of the question and the underlying data schemas, and determined whether the mapping was correct or not and whether any expected mappings were missing. For the *query-generation* stage, the judges looked at each triple pattern and determined whether the pattern was meaningful for the question or not and whether any expected triple pattern was missing. Note that, because our approach does not use any query templates, the same question may generate semantically equivalent queries that differ widely in terms of their structure. Hence, we rely on our evaluation metrics that are based on triple patterns, as there is no gold-standard query for a given question. For the *query-answering* stage, the judges were asked to identify if the result sets for the generated queries are satisfactory.

With these assessments, we computed overall quality measures by both micro-averaging and macro-averaging. Micro-averaging aggregates over all assessed items (e.g., q-node/s-node pairs or triple patterns) regardless of the questions to which they belong. Macro-averaging first aggregates the items for the same question, and then averages the quality measure over all questions.

For a question q and item set s in one of the stages of evaluation, let $correct(q, s)$ be the number of correct items in s, $ideal(q)$ be the size of the ideal item set and $retrieved(q, s)$ be the number of retrieved items, we define coverage and precision as follows:

$$cov(q, s) = correct(q, s)/ideal(q)$$
$$prec(q, s) = correct(q, s)/retrieved(q, s).$$

5.3 Results & Discussion

5.3.1 Disambiguation

Table 1 shows the results for disambiguation in terms of macro and micro coverage and precision. For both datasets, coverage is high as few mappings are missing. We obtain perfect precision for QALD-1 as no mapping that we generate is incorrect, while for NAGA we generate few incorrect mappings.

5.3.2 Query Generation

Table 2 shows the same metrics for the generated triple patterns. The results are similar to those for disambiguation. Missing or incorrect triple patterns can be attributed to (i) incorrect mappings in the disambiguation stage or (ii) incorrect detection of dependencies between phrases despite having the correct mappings.

5.3.3 Question Answering

Table 3 shows the results for query answering. Here, we attempt to generate answers to questions by executing the generated queries over the datasets. The table shows the number of questions for which the system successfully generated SPARQL queries (#queries), and among those, how many resulted in satisfactory answers as judged by our evaluators (#satisfactory). Answers were considered unsatisfactory when: 1) the generated SPARQL query was wrong, 2) the result set was empty due to the incompleteness of the underlying knowledge base, or 3) a small fraction of the result set was relevant to the question. For both sets of questions, most of the queries that were perceived unsatisfactory were ones that returned no answers. Table 4 shows a set of example QALD questions, the corresponding SPARQL queries and sample answers.

Benchmark	QALD-1	NAGA
cov_{macro}	0.973	0.934
$prec_{macro}$	1.000	0.934
cov_{micro}	0.963	0.945
$prec_{micro}$	1.000	0.941

Table 1: Disambiguation

Benchmark	QALD-1	NAGA
cov_{macro}	0.975	0.894
$prec_{macro}$	1.000	0.941
cov_{micro}	0.956	0.847
$prec_{micro}$	1.000	0.906

Table 2: Query generation

Benchmark	QALD-1	NAGA
#questions	27	44
#queries	20	41
#satisfactory	10	15
#relaxed	+3	+3

Table 3: Query answering

Question	Generated Query	Sample Answers
1. Who was the wife of President Lincoln?	`?x marriedTo Abraham_Lincoln . ?x type person`	`Mary_Todd_Lincoln`
2. In which films did Julia Roberts as well as Richard Gere play?	`?x type movie . Richard_Gere actedIn ?x . Julia_Roberts actedIn ?x`	`Runaway_Bride` `Pretty_Woman`
3. Which actors were born in Germany?	`?x type actor . ?x bornIn Germany`	NONE

Table 4: Example questions, the generated SPARQL queries and their answers

Queries that produced *no* answers, such as the third query in Table 4 were further relaxed using an incarnation of the techniques described in (Elbassuoni et al., 2009), by retaining the triple patterns expressing type constraints and relaxing all other triple patterns. Relaxing a triple pattern was done by replacing all entities with variables and casting entity mentions into keywords that are attached to the relaxed triple pattern. For example, the QALD question "Which actors were born in Germany?" was translated into the following SPARQL query: `?x type actor . ?x bornIn Germany` which produced no answers when run over the Yago2 knowledge base since the relation `bornIn` relates people to cities and not countries in Yago2. The query was then relaxed into: `?x type actor . ?x bornIn ?z[Germany]`. This relaxed (and keyword-augmented) triple-pattern query was then processed the same way as triple-pattern queries without any keywords. The results of such query were then ranked based on how well they match the keyword conditions specified in the relaxed query using the ranking model in (Elbassuoni et al., 2009). Using this technique, the top ranked results for the relaxed query were all actors born in German cities as shown in Table 5.

After relaxation, the judges again assessed the results of the relaxed queries and determined whether they were satisfactory or not. The number of additional queries that obtained satisfactory answers after relaxation are shown under #relaxed in Table 3.

The evaluation data, in addition to a demonstration of our system (Yahya et al., 2012), can be found at `http://mpi-inf.mpg.de/yago-naga/deanna/`.

6 Related Work

Question answering has a long history in NLP and IR research. The Web and Wikipedia have proved to be a valuable resource for answering fact-oriented questions. State-of-the-art methods (Hirschman and Gaizauskas, 2001; Kwok et al., 2001; Zheng, 2002; Katz et al., 2007; Dang et al., 2007; Voorhees, 2003) cast the user's question into a keyword query to a Web search engine (perhaps with phrases for location and person names or other proper nouns). Key to finding good results is to retrieve and rank sentences or short passages that contain all or most keywords and are likely to yield good answers. Together with trained classifiers for the question type (and thus the desired answer type), this methodology performs fairly well for both factoid and list questions.

IBM's Watson project (Ferrucci et al., 2010) demonstrated a new kind of *deep QA*. A key element in Watson's approach is to decompose complex questions into several cues and sub-cues, with the aim of generating answers from matches for the various cues (tapping into the Web and Wikipedia). Knowledge bases like DBpedia (Auer et al., 2007), Freebase (Bollacker et al., 2008), and Yago (Suchanek et al., 2007)) are used for both answering parts of questions that can be translated to structured form (Chu-Carroll et al., 2012) and type-checking possible answer candidates and thus filtering out spurious results (Kalyanpur et al., 2011).

The recent QALD-1 initiative (QAL, 2011) proposed a benchmark task to translate questions into SPARQL queries over linked-data sources like DBpedia and MusicBrainz. FREyA (Damljanovic et al., 2011), the best performing system, relies on

Q: ?x type actor . ?x wasBornIn ?z[Germany]
Martin_Lawrence type actor . Martin_Lawrence wasBornIn Frankfurt_am_Main
Robert_Schwentke type actor . Robert_Schwentke wasBornIn Stuttgart
Willy_Millowitsch type actor . Willy_Millowitsch wasBornIn Cologne
Jerry_Zaks type actor . Jerry_Zaks wasBornIn Stuttgart

Table 5: Top-4 results for the QALD question "Which actors were born in Germany?" after relaxation

interaction with the user to interpret the question. Earlier work on mapping questions into structured queries includes the work by Frank et al. (2007) and Unger and Cimiano (2011). Frank et al. (2007) used lexical-conceptual templates for query generation. However, this work did not address the crucial issue of disambiguating the constituents of the question. In Pythia, Unger and Cimiano (2011) relied on an ontology-driven grammar for the question language so that questions could be directly mapped onto the vocabulary of the underlying ontology. Such grammars are obviously hard to craft for very large, complex, and evolving knowledge bases. Nalix is an attempt to bring question answering to XML data (Li, Yang, and Jagadish, 2007) by mapping questions to XQuery expressions, relying on human interaction to resolve possible ambiguity.

Very recently, Unger et al. (2012) developed a template-based approach based on Pythia, where questions are automatically mapped to structured queries in a two step process. First, a set of query templates are generated for a question, independent of the knowledge base, determining the structure of the query. After that, each template is instantiated with semantic items from the knowledge base. This performs reasonably well for the QALD-1 benchmark: out of 50 test questions, 34 could be mapped, and 19 were correctly answered.

Efforts on user-friendly exploration of structured data include keyword search over relational databases (Bhalotia et al., 2002) and structured keyword search (Pound et al., 2010). The latter is a compromise between full natural language and structured queries, where the user provides the structure and the system takes care of the disambiguation of keyword phrases.

Our joint disambiguation method was inspired by recent work on NED (Milne and Witten, 2008; Kulkarni et al., 2009; Hoffart et al., 2011a) and WSD (Navigli, 2009). In contrast to this prior work on related problems, our graph construction and constraints are more complex, as we address the joint mapping of arbitrary phrases onto entities, classes, or relations. Moreover, instead of graph algorithms or factor-graph learning, we use an ILP for solving the ambiguity problem. This way, we can accommodate expressive constraints, while being able to disambiguate all phrases in a few seconds.

DEANNA uses dictionaries of names and phrases for entities, classes, and relations. Spitkovsky and Chang (2012) recently released a huge dictionary of pairs of phrases and Wikipedia links, derived from Google's Web index. For relations, Nakashole et al. (2012) released PATTY, a large taxonomy of patterns with semantic types.

7 Conclusions and Future Work

We presented a method for translating natural-language questions into structured queries. The novelty of this method lies in modeling several mapping stages as a joint ILP problem. We harness type signatures and other information from large-scale knowledge bases. Although our model, in principle, leads to high combinatorial complexity, we observed that the Gurobi solver could handle our judiciously designed ILP very efficiently. Our experimental studies showed very high precision and good coverage of the query translation, and good results in the actual question answers.

Future work includes relaxing some of the limitations that our current approach still has. First, questions with aggregations cannot be handled at this point. Second, queries sometimes return empty answers although they perfectly capture the original question, but the underlying data sources are incomplete or represent the relevant information in an unexpected manner. We plan to extend our approach of combining structured data with textual descriptions, and generate queries that combine structured search predicates with keyword or phrase matching.

References

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

Bhalotia, G.; Hulgeri, A.; Nakhe, C.; Chakrabarti, S.; and Sudarshan, S. 2002. Keyword Searching and Browsing in Databases using BANKS. In *ICDE*.

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

Chu-Carroll, J.; Fan, J.; Boguraev, B. K.; Carmel, D.; and Sheinwald, D.; Welty, C. 2012. Finding needles in the haystack: Search and candidate generation. In *IBM J. Res. & Dev., vol 56, no.3/4*.

Damljanovic, D.; Agatonovic, M.; and Cunningham, H. 2011. *FREyA: an Interactive Way of Querying Linked Data using Natural Language*.

Dang, H. T.; Kelly, D.; and Lin, J. J. 2007. Overview of the trec 2007 question answering track. In *TREC*.

de Marneffe, M. C.; Maccartney, B.; and Manning, C. D. 2006. Generating typed dependency parses from phrase structure parses. In *LREC*.

Elbassuoni, S.; Ramanath, M.; Schenkel, R.; Sydow, M.; and Weikum, G. 2009. Language-model-based ranking for queries on rdf-graphs. In *CIKM*.

Elbassuoni, S.; Ramanath, M.; and Weikum, G. 2011. Query relaxation for entity-relationship search. In *ESWC*.

Fader, A.; Soderland, S.; and Etzioni, O. 2011. Identifying relations for open information extraction. In *EMNLP*.

Ferrucci, D. A.; Brown, E. W.; Chu-Carroll, J.; Fan, J.; Gondek, D.; Kalyanpur, A.; Lally, A.; Murdock, J. W.; Nyberg, E.; Prager, J. M.; Schlaefer, N.; and Welty, C. A. 2010. Building Watson: An Overview of the DeepQA Project. *AI Magazine* 31(3).

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

Gurobi Optimization, Inc. 2012. Gurobi Optimizer Reference Manual. http://www.gurobi.com/.

Heath, T., and Bizer, C. 2011. *Linked Data: Evolving the Web into a Global Data Space*. San Rafael, CA: Morgan & Claypool, 1 edition.

Hirschman, L., and Gaizauskas, R. 2001. Natural Language Question Answering: The View from Here. *Nat. Lang. Eng.* 7.

Hoffart, J.; Mohamed, A. Y.; Bordino, I.; Fürstenau, H.; Pinkal, M.; Spaniol, M.; Taneva, B.; Thaterm S.; and Weikum, G. 2011. Robust Disambiguation of Named Entities in Text. In *EMNLP*.

Hoffart, J.; Suchanek, F. M.; Berberich, K.; Lewis-Kelham, E.; de Melo, G.; and Weikum, G. 2011. Yago2: exploring and querying world knowledge in time, space, context, and many languages. In *WWW (Companion Volume)*.

Kalyanpur, A.; Murdock, J. W.; Fan, J.; and Welty, C. A. 2011. Leveraging community-built knowledge for type coercion in question answering. In *International Semantic Web Conference*.

Katz, B.; Felshin, S.; Marton, G.; Mora, F.; Shen, Y. K.; Zaccak, G.; Ammar, A.; Eisner, E.; Turgut, A.; and Westrick, L. B. 2007. CSAIL at TREC 2007 Question Answering. In *TREC*.

Kulkarni, S.; Singh, A.; Ramakrishnan, G.; and Chakrabarti, S. 2009. Collective annotation of wikipedia entities in web text. In *KDD*.

Kwok, C. C. T.; Etzioni, O.; and Weld, D. S. 2001. Scaling Question Answering to the Web. In *WWW*.

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

Milne, D. N., and Witten, I. H. 2008. Learning to link with wikipedia. In *CIKM*.

Ndapandula Nakashole, Gerhard Weikum and Fabian Suchanek 2012. PATTY: A Taxonomy of Relational Patterns with Semantic Types. In *EMNLP*.

Navigli, R. 2009. Word sense disambiguation: A survey. *ACM Comput. Surv.* 41(2).

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

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

Resnik, P. 1995. Using Information Content to Evaluate Semantic Similarity in a Taxonomy. In *IJCAI*.

Spitkovsky, V. I. Spitkovsky; Chang, A. X. ; 2012. A Cross-Lingual Dictionary for English Wikipedia Concepts. In *LREC*.

Suchanek, F. M.; Kasneci, G.; and Weikum, G. 2007. Yago: a core of semantic knowledge. In *WWW*.

Tummarello, G.; Cyganiak, R.; Catasta, M.; Danielczyk, S.; Delbru, R.; and Decker, S. 2010. Sig.ma: Live views on the web of data. *J. Web Sem.* 8(4).

Unger, C.; and Cimiano, P. 2011. Pythia: Compositional Meaning Construction for Ontology-Based Question Answering on the Semantic Web. In *NLDB*.

Unger, C.; Bühmann, L.; Lehmann, J.; Ngonga Ngomo, A.-C.; Gerber, D.; and Cimiano, P. 2012. Template-based question answering over RDF data. In *WWW*.

Voorhees, E. M. 2003. Overview of the trec 2003 question answering track. In *TREC*.

Yahya, M.; Berberich, K.; Elbassuoni, S.; Ramanath, M.; Tresp, V.; and Weikum, G. 2012. Deep answers for naturally asked questions on the web of data. In *WWW*.

Zheng, Z. 2002. AnswerBus Question Answering System. In *HLT*.