Robust Question Answering over the Web of Linked Data

Mohamed Yahya, Klaus Berberich,
Shady Elbassuoni†, Gerhard Weikum

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

†American University of Beirut, Lebanon
se58@aub.edu.lb

ABSTRACT

Knowledge bases and the Web of Linked Data have become important assets for search, recommendation, and analytics. Natural-language questions are a user-friendly mode of tapping this wealth of knowledge and data. However, question answering technology does not work robustly in this setting as questions have to be translated into structured queries and users have to be careful in phrasing their questions. This paper advocates a new approach that allows questions to be partially translated into relaxed queries, covering the essential but not necessarily all aspects of the user's input. To compensate for the omissions, we exploit textual sources associated with entities and relational facts. Our system translates user questions into an extended form of structured SPARQL queries, with text predicates attached to triple patterns. Our solution is based on a novel optimization model, cast into an integer linear program, for joint decomposition and disambiguation of the user question. We demonstrate the quality of our methods through experiments with the QALD benchmark.

Categories and Subject Descriptors

H.3.3 [**Information systems**]: Information Search and Retrieval; I.2.1 [**Artificial Intelligence**]: Natural language interfaces

Keywords

question answering; knowledge base; semantic search; disambiguation; usability

1. INTRODUCTION

Motivation: With the success of IBM's Watson system [23], natural-language question answering (QA) is being revived as a key technology towards coping with the deluge of digital contents. Watson primarily tapped into textual contents, with limited use of structured background knowledge. However, there is a strongly growing wealth of structured data on the Web: large knowledge bases like `dbpedia.org`, `freebase.com`, and `yago-knowledge.org`; billions of inter-linked RDF triples from these and other sources, together forming the Web of Linked Data [21]; hundreds of millions of Web tables (e.g., accessible via `research.google.com/tables`); and a large amount of RDF-style microdata embedded in HTML pages (e.g., using the `schema.org` vocabulary).

This paper is about conveniently and effectively searching this wealth of structured Web data, specifically, the RDF-based world of Linked Data. Although there are structured query languages like SPARQL that could, in principle, be used to this end, such an approach is impractical for several reasons: i) users are not familiar with the diverse vocabulary of the data, ii) the data itself exhibits high heterogeneity so that even a power-user would struggle with formulating the right queries, iii) even a perfectly formulated query may fail to find answers if the query vocabulary (predicates, classes, entity names) does not match the data vocabulary. To bridge the gap between users' information needs and the underlying data and knowledge, it has recently been proposed to take natural-language questions as user input and automatically translate these into structured queries, for example, in the SPARQL language [40, 46, 35].

Example: Consider the question "Which music bands covered songs written by the Rolling Stones?", or in conveniently short form: "bands covering songs by the Stones", or in even more telegraphic style: "bands songs Stones". An ideal QA system for RDF data would automatically translate this user input into a SPARQL query with the following triple patterns:

```
?x type MusicBand .
?x performed ?y .
?y type Song .
RollingStones created ?y .
```

where `?x` and `?y` are variables, and the binding for `?x` is the query answer. Getting this translation right is all but an easy task. Moreover, even this seemingly perfect query may not work if the underlying data differs from the vocabulary and structure of the query. For example, the class `Song` may

CIKM 2013 San Francisco, California, USA
ACM 978-1-4503-2263-8/13/10 ...$15.00.
http://dx.doi.org/10.1145/2505515.2505677 .

not be sufficiently populated; instead the data could make more frequent use of the type `Music`. Similarly, the predicate `created` may be used only for books, paintings, etc., and a different predicate `composed` could be prevalent for music. Finally, it could be that only individual people are listed as composers (rather than bands), so that one would need additional/alternative triple patterns:

```
?z composed ?y and
?z memberOf RollingStones .
```

Problem: Our goal in this paper is to improve QA over RDF data in terms of robustness, by devising a method for generating good queries that return answers even in such hard situations. The solution must not come at the expense of increasing the burden on the user (e.g., by a tedious process of providing multiple question formulations). We would like to increase the system's robustness also by allowing users to be informal if not sloppy in the way they express questions. For example, a variant of our example question could be: "woman song-writer covered love ballads by the stones". A solution must identify the right sub-phrases like "woman song-writer", "covered", "love ballad", and "the stones" and must automatically map them onto semantic targets of the underlying data: types/classes, relations, and entities.

Approach: This paper presents a solution based on generating a set of query variants that cover the user question to different extents. Thus, we do not need a perfect translation, yet we can execute a query and return good answers. We still face the problem that the noun phrases and verbal phrases of the input are highly ambiguous and can be mapped to different semantic targets. We address this issue by a joint optimization for i) identifying the boundaries of the relevant phrases (e.g., "woman song-writer" vs. "song-writer") and ii) jointly mapping all or a subset of these phrases onto classes, relations, and entities. The optimization is cast into an integer linear program with an appropriately designed objective function and constraints.

Sometimes a better query is generated by leaving out intractable parts of the user question. For example, "love ballad" may not map to any class and even "woman song-writer" may be too hard so that simply mapping it to the class `Musician` may be the best option. To compensate for such incomplete translations, we tap into textual descriptions that are often available with structured Web data. These could be sources from which RDF triples have been gathered, texts about entities from their Wikipedia articles or homepages, or the contents that surround a table or microdata in a Web page. We extend SPARQL triple patterns into the notion of *SPOX quad patterns* that consist of the usual subject-predicate-object (SPO) parts and a teXt component with keywords or text phrases that need to be matched by the textual descriptions associated with the SPO triples in the data. For the example question, we could generate a query with the quad patterns:

```
?x type singer {"woman", "song-writer"} .
?x performed ?y .
?y type Music {"love ballad"} .
RollingStones ?p ?y {"by"} .
```

Contribution: This paper makes the following contributions: i) extending the translation of natural-language questions into SPARQL queries by means of quad patterns with textual conditions; ii) a model that allows generating relaxed queries that capture only parts of the user inputs in SPO conditions and map other parts onto text conditions; iii) an optimization method based on an integer linear program that identifies the phrases to be mapped and computes a joint mapping for disambiguation; iv) a strategy for generating relaxed queries that return many answers and a suitable run-time environment. and v) ultimately, improving the robustness of question-to-query translation without impeding user convenience.

2. COMPUTATIONAL MODEL

RDF data and text: The Linked Data sources that we operate on consist of subject-predicate-object (SPO) triples following the RDF data model. Example triples are:

```
KeithRichards composed Angie .
Womack&Womack performed Angie .
Angie type Music .
```

As such data is often extracted or compiled from text-rich Web pages such as Wikipedia articles or thematic Web portals such as `last.fm`, we associate with each triple a *textual context*, thus forming SPOX quadruples, quads for short. An example quad could be

```
Angie type Music {"...a ballad which tells of the
end of a romance ..."} .
```

For Linked Data and for Web tables, such text extensions of structured data are very natural and can be easily compiled.

We can query this kind of data+text combination by extending SPO triple patterns, the basic building block of the SPARQL query language, with a search condition about keywords or phrases: `?x type song {"love", "ballad"}`. The semantics of a quad pattern is that a data triple must satisfy the SPO condition and its associated text context should match (at least partially) the specified text condition.

Input questions: We interpret the user's input question as a sequence of phrases, where each phrase is a sub-sequence of consecutive words. We run a part-of-speech (POS) tagger and a dependency parser on the question. This way, we generate a set of candidates for sub-phrases that could be mapped to semantic items like classes, relations, or entities. The candidate phrases can overlap; our optimization model takes care of selecting a consistent subset of phrases (see Section 3).

Phrases potentially corresponding to classes and entities are detected using a large dictionary of surface forms. POS tags guide the identification of candidate relation phrases. We rely on an extension of the relational patterns in [16] and extend these with common nouns from a dictionary. This approach produces heavily overlapping phrases. However, we only consider consecutive words, to avoid speculating about user intentions and to be robust against informal formulations. We include long phrases, the rationale being that these can sometimes be directly mapped to specific classes in the data (e.g., matching Wikipedia category names).

Predicate-argument dependencies: Some phrases will eventually be mapped to relations, so it is important to identify their arguments. We restrict ourselves to binary relations and aim to identify left/right arguments based on

the dependency parsing of the input question. For each of the candidate phrases that could possibly denote a relation, we check for words that have particular dependencies with words in the phrase. The words that these dependencies point to are identified using the Stanford dependency parser [29], and all pairs of phrases that contain these words become candidates for left/right arguments of the potential-relation phrase. This is relatively liberal, as the considered dependencies can originate at different words of the phrase. The rationale here is to minimize the risk that we miss relevant dependencies; for selecting the correct argument structure, we rely on the optimization model (see Section 3).

Dependency parsing is essential for questions that refer to multiple relationships. An example is "bands from the US who covered songs by the Stones and the Beatles". It is vital to recognize that the phrase "covered" has the left-hand argument "bands" and two right-hand arguments "songs by the Stones" and "songs by the Beatles".

Because we rely on translating questions to a triple-based language, noun phrases can often stand for a relation and its argument. An example is the question "songs by the Stones". Conceptually, the phrase "songs by" expresses both a relation and a class (the answer type). If the phrase "songs by" is mapped to the relation `performed`, then we also add a *latent concept phrase* to stand for one of the two arguments of the relation. This is needed since our optimization model (see Section 3) constrains each phrase to map to at most one semantic target in the data.

Additionally, some patterns, such as adjective forms of countries and regions, and prepositions, can denote the existence of a relation that is unspecified. For example, a question asking for "Swedish skateboarders" indicates the existence of a relation between `Sweden` and one or more members of the class `Skateboarder`. This relation is not specified in any token, so we generate a *latent relation phrase* to account for this, which, if chosen, would correspond to a wild card relation `?r`. During query processing `?r` will be bound to relations such as `bornIn`, `livesIn`, etc. It is possible that the knowledge has a class `SwedishSkateboarders`, the optimization model would make the decision which of the two interpretations of "Swedish skateboarders" to choose.

Answer type: To determine the output structure (corresponding to the Select clause) for the query to be generated, we need to infer the type of the answer(s) that the user expects, based on the question formulation. We use a sequence of heuristics: i) if a question word like "who", "where", etc. or "which" with a modifier is present, it determines the answer type (i.e., person, location, etc., or the type of the modifier); ii) without such words, the head noun of the question's subject determines the output type (e.g., "bands" in "bands from the US covering ...") unless this is determined by the disambiguation model to be part of an entity phrase, in which case iii) the first common noun occuring in the sentence that maps to a class determines the answer type.

Output: The main task addressed in this paper is to select a subset of candidate phrases and map them onto semantic target items, where possible targets are classes, relations, and individual entities present in the underlying data and knowledge sources. We assume that we have a comprehensive dictionary of semantic items compiled from the data. For a given phrase, the candidate targets are those semantic items whose names or textual description (e.g., synonyms, glosses, or salient keyphrases) overlap with the wording in the phrase. For example, "the stones" would have candidates like gemstones (a class), the Rolling Stones (an entity of type `MusicBand`), etc. The candidates for the phrase "cover" would include book covers (class), album covers (class), perform (relation, between musician and song), treat a subject (relation, e.g., between books and events), Thomas M. Cover (entity), etc. Compared to many other disambiguation tasks (e.g., for entity names), a major complexity faced here is the wide range of possible interpretations where not even the kind of semantic item is a priori clear.

More formally, given:

(i) a set $P = \{p_1, p_2, \dots\}$ of possibly overlapping candidate phrases,

(ii) a set $S = \{s_1, s_2, \dots\}$ of semantic target candidates as input, and

(iii) a set of phrase dependencies $D_P = \{p_{rel_1}(p_{subj_1}, p_{obj_1}), p_{rel_2}(p_{subj_2}, p_{obj_2}), ...\}$,

the desired output is:

(i) a selected subset $P^* = \{p_{i_1}, p_{i_2}, \dots\} \subseteq P$ of phrases,

(ii) a functional mapping $P^* \rightarrow S$, and

(iii) a set of semantic target dependencies $D_S = \{s_{rel_1}(s_{subj_1}, s_{obj_1}), s_{rel_2}(s_{subj_2}, s_{obj_2})...\}$

that satisfy certain constraints.

We will discuss this task in Section 3. It is important to note that the mapping can be partial regarding P: not every phrase needs to be mapped (even if it does not overlap with other phrases).

3. OPTIMIZATION MODEL FOR JOINT DECOMPOSITION AND DISAMBIGUATION

We design an integer linear program (ILP) to jointly resolve the question decomposition and disambiguation problems. Our model couples the selection of phrases and their mapping onto semantic targets. Importantly, we introduce constraints that ensure that phrases are selected in a way that preserves their phrase dependencies in the image of the mapping onto semantic targets. This guarantees that we arrive at a set of chosen semantic items that are naturally grouped into triples and thus yield a well-formed SPARQL query.

In addition to the sets P, S, D_P and D_S introduced in the previous section, our model makes use of two kinds of pre-computed weights: (i) $s(i,j)$ denotes a prior score for phrase p_i mapping to semantic target s_j, regardless of the context, and (ii) $r(k,l)$ denotes the semantic relatedness between semantic target items k and l, based on co-occurrences in the underlying data and knowledge sources (Yago, DBpedia, Wikipedia), to integrate the question context in scoring.

We now define the variables of the ILP, all of which are 0/1 decision variables:

- X_i indicates if phrase p_i is selected.
- $Y_{i,j}$ indicates if the mapping $p_i \mapsto s_j$ is chosen.
- Z_k is 1 if $s_k \in S$ appears in the image of the chosen mapping.

- $Z_{k,l}$ is 1 if both Z_k and Z_l are set to 1.
- $Q_{m,n,d}$ indicates if the phrase p_n is chosen as part of the phrase dependency $d_m \in D_p$, in role $d \in \{rel, subj, obj\}$.
- T_t indicates for semantic dependency $s_{rel_t}(s_{subj_t}, s_{obj_t})$ if all three components are chosen, that is, Z_{rel_t}, Z_{subj_t}, Z_{obj_t} are 1, *and* the corresponding SPARQL triple pattern has a non-empty result in the underlying data.

The result of the ILP is a 0/1 assignment of the X, Y, Z, Q, and T variables, from which a mapping $P^* \to S$ and a set of semantic dependencies D_S can be read off. The Q and T variables couple the choice of phrases and their mapping to semantic targets with the dependencies among phrases and semantic items. This will ensure that the output consists of meaningful triples, rather than mapping phrases to targets independently. Moreover, the T variables also encode if a semantic dependency in the output actually produces answers when the corresponding triple pattern (alone) were executed on the data. The objective function below rewards decisions that lead to non-empty answers.

The following 0/1 constants are also used:

- C_j, E_j, and R_j are 0/1 constants indicating that the semantic target s_j is a class, entity, or relation, respectively, where $\sum_j C_j + E_j + R_j = 1$.
- t_{rc} indicates if the semantic relation s_r and the concept s_c are type compatible.

$\mathcal{P}(t)$ is the set of all phrases that contain the token (word) t. The set of latent concept phrases is $\mathcal{P}_{lat}$, and each latent concept phrase $p_{lat} \in \mathcal{P}_{lat}$ phrase is generated by a relation phrase $p_r = gen(p_{lat})$.

Objective function: The objective of the ILP is to maximize the following function:

$$\alpha \sum_{i,j} s(i,j) Y_{i,j} + \beta \sum r(k,l) Z_{k,l} + \gamma \sum_{m,n,d} Q_{m,n,d} + \delta \sum_t T_t$$

The first term aims for good phrase-target mappings if each phrase were mapped separately. This is balanced against the second term which rewards mappings that result in two highly related entities being together in the mapping image. The third term reflects the goal of capturing phrase dependencies. The fourth term rewards decisions that lead to triple patterns with non-empty answers. The coefficients α, β, γ, δ are hyper-parameters to be tuned with a small set of withheld training data, that is, pairs of questions and good query translations.

Constraints: The model described until now can be seen as a compact way to encode all possible interpretations of a questions with respect to a knowledge base. Some interpretations do not make sense, such as those where a type constraint is violated or a word is part of multiple phrases (taking into consideration latent ones). Other interpretations, such as those that cannot be answered by the knowledge base, should be less favored, everything else beging equal. The optimization is subject to the following constraints which both forbid nonsensical interpretations (1-12), and downweigh empty interpretations of a query (13, 14):

1. A phrase maps to at most one semantic target.
$$\sum_j Y_{ij} \leq 1, \forall i$$
2. If a mapping $p_i \mapsto s_j$ is chosen, then the target node must be chosen
$$Y_{ij} \leq X_i, \forall j$$
3. If a mapping $p_i \mapsto s_j$ is chosen, then no phrase that overlaps with p_i can be chosen.
$$\sum_{i \in \mathcal{P}(t)} X_i \leq 1, \forall t \in T$$
4. $Z_{k,l}$ is 1 iff both Z_k and Z_l are 1.
$$Z_{k,l} + 1 = Z_k + Z_l, \forall Z_{k,l}$$
5. X_k is 1 and Z_l is 1 iff $Y_{k,l}$ is 1.
$$Y_{k,l} + 1 = X_k + Z_l, \forall Y_{k,l}$$
6. Each semantic triple can contribute to each role once at most.
$$\sum_{Q_{mnd}} \leq 1, \forall m, d$$
7. If $Q_{m,n,d} = 1$ then the corresponding p_n must be selected ($X_n = 1$).
$$Q_{mnd} \leq X_n, \forall m, d$$
8. Each chosen phrase dependency (encoded in the values of the Q variables) must include a relation phrase.
$$E_r \geq Q_{mn'd} + X_{n'} + Y_{n'r} - 2, \forall m, n', r, d = rel$$
9. Each semantic triple should have at least one class to join with other semantic triples.
$$\begin{aligned} C_{c_1} + C_{c_2} \geq \; & Q_{mn''d_1} + X_{n''} + Y_{n'''c_1} + \\ & Q_{mn'''d_2} + X_{n'''} + Y_{n'''c_2} - 5, \\ \forall m, n'', n''', r, c_1, c_2, d_1 = arg1, d_2 = arg2 & \end{aligned}$$
10. If any two Q variables have a token as part of two different mentions, at most one of the two can be chosen.
$$\begin{aligned} & Q_{mnd} + Q_{m'n'd'} \leq 1, \\ & \forall n, n', d, d', m \neq m', t \in p_n, t \in p_{n'}, p_n \neq p_{n'} \end{aligned}$$
11. Each relation in a chosen semantic dependency (encoded in values of the Q and Y variables) must have a type signature that is compatible with the types of its left and right arguments (classes or entities) in the semantic dependency.
$$\begin{aligned} t_{rc_1} + t_{rc_2} \geq \; & Q_{mn'd_1} + X_{n'} + Y_{n'r} + \\ & Q_{mn''d_2} + X_{n''} + Y_{n'''c_1} + \\ & Q_{mn'''d_3} + X_{n'''} + Y_{n'''c_2} - 7 \\ \forall m, n', n'', n''', r, c_1, c_2, & \\ d_1 = rel, d_2 = arg1, d3 = arg2 & \end{aligned}$$
12. A latent concept phrase p_l (see Section 2) can be selected only if the generating relation phrase p_r is also selected.
$$X_l \leq X_r, \forall p_l \in \mathcal{P}_{lat}, p_r = gen(p_l)$$

13. If a T variable is 1 then all variables X, Y, Z and Q variables in the semantic dependency it encodes are chosen.

$$T_t \leq X_i, \forall t, X_i, s.t.\ p_i \mapsto s_j, s_j \text{ part of } T_t$$

$$T_t \leq Z_k, \forall t, Z_k, s.t.\ s_k \text{ part of } T_t$$

$$T_t \leq Y_{ij}, \forall t, Y_{ij}, s.t.\ p_i \mapsto s_j, s_j \text{ part of } T_t$$

$$T_t \leq Q_{tid}, \forall t, i, d, s.t.\ p_i \mapsto s_{j_d}, s_{j_d} \text{ part of } T_t \text{ in role } d$$

14. T variables corresponding to triple patterns that have no matches in the data are set to 0.

$$T_t = 0, \forall s_{rel_t}(s_{subj_t}, s_{obj_t}) \text{ with no matches in KB.}$$

It is worth noting that the last constraint decreases the score of an interpretation in a triple pattern with no results in the KB, without forbidding such interpretations.

By iteratively adding a constraint that prevents the previous solution to the ILP (corresponding to an interpretation of the question), we are able to generate multiple interpretations of a question in descending order of their scores.

4. QUERY GENERATION

Much of the query to be generated will naturally fall out from the computed mapping $P^* \rightarrow S$ and the semantic dependencies D_S. As we also identify a phrase for the answer type (Section 2), we could now directly produce a set of triple-pattern conditions and a Select clause for a full-fledged SPARQL query. To ensure that the triple patterns join in a proper manner (through shared variables), we substitute each class c by a variable `?c` and add a triple pattern `?c type c`. For example, with the mappings "*bands*" $\mapsto$ `MusicBand`, "*cover*" $\mapsto$ `performed`, "*songs*" $\mapsto$ `Music`, we would first obtain the triple pattern `MusicBand performed Music` and then expand it into:

```
?var1 performed ?var2 .
?var1 type MusicBand .
?var2 type Music.
```

As the mapping of phrases to classes or relations often comes with a semantic generalization step (e.g., mapping "love ballad" to `Song` or `Music`), we may even consider already mapped phrases as candidates for additional text conditions. Finally, even if the generated query captures most or all of the input question, it may be overly specific and does not necessarily return answers. Next we discuss how to overcome this issue.

5. QUERY EXTENSION AND RELAXATION

Structured and keyword querying are two paradigms traditionally kept apart. We argue that a combination of both works best in our QA setting. We combine them by generating SPOX quads rather than pure-SPARQL triple patterns only. We start with the SPO query that is generated from the outcome of the ILP-based optimization as explained in Section 4. Then we apply three kinds of extension and relaxation techniques.

Text extension: The first technique identifies phrases or words in phrases that are not mapped to any of the triple patterns in the generated query. Some words may not be part of any detected phrase. Others may belong to detected phrases that, in the final disambiguation, could not be placed into a semantic triple pattern, and are hence not part of the structured query. Finally, it is possible that triple patterns are generated but are not connected to the output variable (Select clause) of the final query via join operations. To avoid computing Cartesian products and hard-to-interpret results, such triple patterns are discarded leading to "left-over" phrases. In all three cases, the words are attached as keywords to the triple pattern for the type of the query output.

Empty-result relaxation: The second technique involves triple patterns that lead to empty results when executed against the underlying data. An empty-result SPO condition indicates either a disambiguation error or an overly specific query that lacks coverage in the underlying data due to poorly populated classes or relations. It is also possible that every single SPO condition produces answers, but their combination yields an empty result. An example is:

```
?x type Singer .
?x type Woman .
```

We may have many singers and many women in the data, but no or only very few female singers. In such cases, it is desirable to use only one of the two SPO conditions and cast the other one into a text condition of a SPOX quad pattern:

```
?x type Singer {"woman"} .
```

We compute this form of relaxation in an iterative bottom-up manner. The starting point is the single SPO pattern that encodes the answer type: `?x type <class>`. We then iteratively add triple patterns that have a join variable (initially `?x`) in common with previously added triple patterns. We check if the resulting query has an empty result: if so, the last added pattern is removed. This proceeds until there are no more patterns to be added. Words from phrases corresponding to semantic targets that have been removed are then added as keywords to the `?x type <class>` pattern.

Extreme relaxation: The third technique that we consider is to cast the entire question into a text condition with an additional type filter on the result. The latter is derived from the answer-type heuristics (Section 2). An example would be

```
?x type Music {"band", "cover", "love", "ballad",
               "by", "stones"} .
```

The iterative procedure for the result-emptiness relaxation degrades into extreme relaxation if none of the considered queries produces answers. Extreme relaxation is also used when the ILP model does not produce any SPO triple patterns at all. Extremely relaxed queries can still be highly beneficial, most notably, when the query expresses a complex class in the knowledge base. For example, a question asking for "American rock bands that ..." can be answered by

```
?x type AmericanRockMusicBands {"..."} .
```

6. RANKING

Relaxation entails that the query becomes less constrained, returning a larger number of results. This necessitates ranking the answers to make the output user-friendly. We employ

statistical language models [48] to rank query results. Our specific approach, which we describe next, is inspired by Elbassuoni et al. [13, 14].

Formally, a query $Q = (q_1, ...q_n)$ consists of keyword-augmented triple patterns of the form

$$q_i =< s_{q_i}, p_{q_i}, o_{q_i}, \mathbf{w}_{q_i} > \ .$$

Analogously, a result $T = (t_1, ...t_n)$ consists of keyword-augmented triples of the form

$$t_i =< s_{t_i}, p_{t_i}, o_{t_i}, \mathbf{w}_{t_i} > \ .$$

Here, $\mathbf{w}_{q_i}$ and $\mathbf{w}_{t_i}$ are bags of keywords associated with the triple pattern and triple, respectively. We further assume that there is a substitution θ from the variables of Q to resources in the knowledge base such that $\forall i, \theta(q_i) = t_i$.

We use a query-likelihood approach to rank results matching a query, which factors in salience of contained entities as well as textual relevance. We define the probability of generating the query Q from a result T as

$$P(Q \mid T) = \prod_{i=1}^{n} P(q_i \mid t_i) \ ,$$

thus assuming that triple patterns are generated independently. The probability of generating the triple pattern q_i from the corresponding triple t_i in the result is defined as

$$P(q_i \mid t_i) = P(s_{q_i}, p_{q_i}, o_{q_i}, \mid s_{t_i}, p_{t_i}, o_{t_i}) \times P(\mathbf{w}_{q_i} \mid \mathbf{w}_{t_i}) \ .$$

We thus assume, for tractability, that the structured and textual part of triple patterns are generated independently.

For the generation of the structured part, we define

$$P(s_{q_i}, p_{q_i}, o_{q_i} \mid s_{t_i}, p_{t_i}, o_{t_i}) = (1-\beta)\, P(s_{t_i}) + \beta\, P(o_{t_i}) \ .$$

The probabilities $P(s_{t_i})$ and $P(o_{t_i})$ reflect the salience of the subject and object. The parameter $\beta \in [0, 1]$ is set according to whether s_{q_i} and/or o_{q_i} are variables in the triple pattern.

We use a unigram language model for the generation of the textual part and define

$$P(\mathbf{w}_{q_i} \mid \mathbf{w}_{t_i}) = \prod_{v \in \mathbf{w}_{q_i}} P(v \mid \mathbf{w}_{t_i})$$

as the probability of generating the bag of keywords associated with the triple pattern q_i from its counterpart in the keyword-augmented triple t_i.

In our concrete implementation, the probabilities $P(s_{t_i})$ $P(o_{t_i})$ are estimated based on the number of incoming links to the Wikipedia articles of s_{t_i} and o_{t_i}. The bag of keywords $\mathbf{w}_{t_i}$ in the keyword-augmented triple t_i is a concatenation of the documents associated with its subject and object. We associate with every entity from the knowledge base such a document, which consists of its infobox, categories, as well as its keyphrases (i.e., anchor texts of hyperlinks to its Wikipedia article). All probabilities in our model are smoothed taking into account global dataset statistics.

The above model considers a query result as a tuple of triples T that match the query. The final result displayed to the user is a projection of this query result. Duplicates in the final result are filtered out, and we only report the one having the highest query likelihood according to our model.

Intuitively, for a (relaxed) query with keyword-augmented triple patterns, our model returns results that match the (relaxed) structured part of the query in an order that favors results with salient entities and relevant keywords.

7. EXPERIMENTAL EVALUATION

7.1 Setup

Data: We used two prominent sources from the Web of Linked Data: the Yago (`yago-knowledge.org`) and DBpedia (`dbpedia.org`) knowledge bases, together comprising several hundred millions of RDF triples. As they link their entities to Wikipedia URLs, we augment each RDF entity with the textual description from the corresponding article. Our implementation manages this data by storing the RDF parts in PostgreSQL and the text parts in Lucene (with proper linkage across the two).

As Yago and DBpedia entities reference each other via `sameAs` links, our slice of Linked Data combines the rich class system of Yago (with more than 300,000 classes including all of WordNet) with the extensive fact collection of DBpedia (with more than 400 million RDF triples derived from Wikipedia infoboxes). Yago also provides a huge set of name-entity and phrase-entity pairs through its `rdfs:label` relation. This is the basis for our dictionary driving the generation of possible mappings from phrases to semantic targets. For classes as potential targets, we also incorporate WordNet synsets. For mapping potential relation phrases detected relying on POS tag patterns, we use the PATTY collection [30], which provides paraphrases for the (RDF properties) in DBpedia and Yago.

Benchmark: As our main test case, we adopted the benchmark from the 2nd Workshop on Question Answering over Linked Data [35]. QALD-2 consists of 100 natural-language test questions of two different flavors: factoid and list questions (plus 100 withheld questions for training). We discarded questions that require counting or return literals (e.g., numbers) rather than entities as answers. For test questions, this resulted in: a) 19 *factoid questions* that are supposed to return exactly one correct result, and b) 30 *list questions* that produce sets of answers. Examples are "What is the capital of Canada" for the former, and "people who were born in Vienna and died in Berlin" for the latter. The QALD-2 benchmark comes with manually compiled correct answers. In the case of factoid questions, this is our ground truth. In the case of list questions, this is what we refer to as *QALD ground truth*. Some of the methods returned additional answers that are correct but not included in the QALD-2 ground truth. We manually assessed the correctness of these extra answers, establishing an *extended ground truth* answer set.

In addition, we experimented with the 48 telegraphic queries used by Pound et al. [34]. 19 of these queries are not real questions, but are merely entity lookups. They give a description of an entity and ask for disambiguation onto the right entity in Yago/DBpedia/Wikipedia. An example is "guernica picasso". We excluded these lookup queries from the test set. Among the remaining 29 true questions, we disregarded 7 cases that request literals rather than entities as answers. This left us with 22 true questions, further broken down into 16 list questions and 6 factoid questions.

Performance measures: All the methods in our experiments return ranked lists of answers. For factoid questions with single correct answers, we use the established notion of *Mean Reciprocal Rank (MRR)* [8] as our main measure of

quality. In addition, we also report on precision at a cut-off rank of 10.

For list questions, we mainly report the *Normalized Discounted Cumulative Gain (NDCG)* [28] as a measure combining precision and recall with geometrically decreasing weights of ranks. Additionally, we give numbers for precision at different cut-off ranks and for Mean Average Precision (MAP).

Measures with a cut-off k are evaluated by taking into consideration that the system might produce less than k results. This can happen when the generated query is unrelaxed (structured only), or the expected result type has few instances. For example, if for question q a list on n results is returned and $rel(e, q)$ is an indicator function equal to 1 if e is a correct result and 0 otherwise, then $Precision@k$ is computed as follows:

$$Precision@k(q) = \frac{\sum_{i=1}^{min(k,n)} rel(e_i, q)}{min(k, n)}.$$

This is accounted for in all other measures in Tables 1 and 2. Later in this section we will consider recall at a certain cut-off. This is handeled differently, as we will explain.

Methods under comparison: We compare our method against three opponents:

- **SPOX+Relax:** This is our full-fledged method with text extension, relaxation, and ranking based on statistical language models.
- **SPO:** The first opponent and main baseline is the generation and execution of purely structured queries. This corresponds to the method of Yahya et al. [46]. Our implementation contains that prior method as a special case.
- **SPO+Relax:** The second baseline is when we restrict our framework to generate only structured SPO queries, without consideration of text conditions. In this case, the relaxation techniques can still choose to partially cover the question to avoid being over-specific, but this baseline does not include any keyword conditions. In both SPO and SPOX+Relax, our ranking approach will rely on salience, as there are no keywords.
- **KW+Type:** The third and simplest baseline is to cast the entire question into keyword search returning entities, but combine this with a filter on the lexcial type of the answers. This corresponds to enforcing the extreme relaxation in our framework, where all generated queries are of the form `?var type <class> {"kw1", "kw2", ...}`. For this special case, our LM-based ranking is improved by using Lucene's tf-idf-based scores for the keyword matches. The reported measures are based on this better ranking.

Parameters were tuned using the QALD-2 training set.

7.2 Results

Tables 1 and 2 show the results for the QALD-2 list and factoid questions, respectively.

For list questions, the numbers, especially for our main success metric NDCG, show the superiority of the SPOX+Relax method, with an NDCG@10 of 0.51. Both variants with pure SPO queries, which represent the prior state-of-the-art in QA over Linked Data, perform significantly worse, with an NDCG@10 of 0.41. The results for these two variants are consistently close to each other, which shows that not considering keywords from the question in SPO+Relax limits the quality of results.

	NDCG		Precision		MAP
	@10	@100	@10	@100	@100
QALD ground truth					
SPO**X**+Relax	0.51	0.53	0.49	0.46	0.58
SPO+Relax	0.41	0.43	0.46	0.44	0.47
SPO	0.41	0.42	0.46	0.44	0.47
KW+Type	0.24	0.29	0.15	0.10	0.28
Extended ground truth					
SPO**X**+Relax	0.60	0.54	0.60	0.48	0.65
SPO+Relax	0.42	0.42	0.49	0.46	0.50
SPO	0.42	0.41	0.49	0.45	0.46
KW+Type	0.30	0.41	0.23	0.13	0.34

Table 1: Results for QALD-2 list questions

	MRR	Precision @10
SPO**X**+Relax	0.72	0.55
SPO+Relax	0.54	0.50
SPO	0.53	0.50
KW+Type	0.15	0.02

Table 2: Results for QALD-2 factoid questions

These methods often miss important search conditions from the input question, or generate overly specific queries. The SPOX+Relax method, on the other hand, behaves more robustly, because it judiciously chooses between structured and keyword conditions in the query and also avoids quad patterns with insufficient coverage in the data. The KW+Type performed very poorly. Despite the manually specified type constraints, these are still not enough to narrow down the space of possible candidates, more structure was needed for most questions.

As an example, take a list question asking for "Swedish professional skateboarders". We initially map it to the SPO query

```
?x type Skateboarder .
?x ?r1 Sweden .
?x r2 ?y .
?y type Professional .
```

Although the query captures the question properly, it returns no results, as the class `Professional` is sparsely populated. Relaxation results in the SPOX query

```
?x type Skateboarder {"professional"} .
?x ?r1 Sweden .
```

which reurns a satisfactory result.

For factoid questions, the general trends are the same, with SPOX+Relax being the clear winner with an MRR score of 0.72, followed by SPO+Relax which scored 0.54. However, the gains here less pronounced here compared to list questions. SPO already performs well with an MRR of 0.53. For factoid questions, the main issue was answer types that were not properly mapped because a different type, with the same surface form as the one intended, was too

	Precision	Recall	F_1
List			
SPO**X**+Relax@$k = 1$	0.50	0.15	0.23
SPO**X**+Relax@$k = 10$	0.49	0.41	0.45
SPO**X**+Relax@$k = 100$	0.46	0.48	0.47
SPO**X**+Relax@$k = 500$	0.44	0.58	0.50
SemSek	0.28	0.29	0.29
MHE	0.26	0.36	0.30
QAKis	0.15	0.16	0.15
Factoid			
SPO**X**+Relax@$k = 1$	0.68	0.68	0.68
SPO**X**+Relax@$k = 2$	0.61	0.74	0.67
SPO**X**+Relax@$k = 5$	0.58	0.79	0.67
SPO**X**+Relax@$k = 10$	0.55	0.79	0.65
SemSek	0.71	0.78	0.74
MHE	0.52	0.57	0.54
QAKis	0.26	0.26	0.26

Table 3: Comparison to other systems in QALD-2 based on the QALD-2 ground truth

prominent. An example of a factoid question where our system does not return satisfactory results is "Who developed Skype?". The expected answer is an orginization (`SkypeTechnologies`). Our system favors mapping "Who" to the class `person`, which means that even with extreme relaxation, which only preserves the type constraint, the correct answer cannot be retrieved.

We also compared our results against the systems participating in QALD-2: SemSek [2], MHE [41], and QAKis [9]. QALD-2 adopts set-based measures, whereas our system performs ranked retrieval to compensate for relaxation. To make our results comparable, we consider precision and recall at various cut-off thresholds k. We computed recall with respect to the size of the ground truth result set, regardless of k. This generally results in penalizing our system as at most 10 relevant results can be returned when $k = 10$, regardless of the total number of relevant results out there. The official QALD-2 evaluation considers "partially right" answers as good enough [41]. Table 3 shows the results.

For list questions, our system clearly outperforms other systems on all measures. Questions here vary between those that have a couple of answers and those that have more than a hundred. As more results are viewed, there is rapid gain in recall for each cut-off threshold, with little sacrifice of precision, which speaks for the ranking approach.

For factoid questions, our system is outperformed only by SemSek, but the margin is smaller than the gains we make on list questions. The main issue is the same as the one above, namely that of prominent classes taking over the result type. This explains the gap between our results and those of SemSek. Still, overall, our system greatly benefits from falling back onto the excpeted result type for relxation, despite these special cases.

For the telegraphic query workload of Pound et al. [34], SPOX+Relax again turned out to be the best method in our experiments. For factoid questions we achieved an MRR of 0.83, and for list questions a precision@10 of 0.73. These numbers are similar to those reported in [34].

Note, that the results are not directly comparable, as that prior work used an old, smaller version of Yago as its sole data source and reported only the combined performance of all questions regardless of their nature (simple entity lookups, factoid questions, difficult list questions).

Table 4 shows the results we obtained for some questions. The first two questions are based on QALD-2, and the third is from the telegraphic query workload of Pound et al. [34].

For the first two, we show the query generated initially, the relaxed query, the number of results in each of the two ground truths we consider, and the number of relevant results in the top 10 answers with respect to each of the two ground truths (both return more than 10 answers). We discussed the first question earlier. The second one results in a SPOX pattern query that includes a keyword component, as the system could not map the verb "dwelt" to an appropriate relation. This query returns satisfactory results to the user.

For the last query, despite the fact that the query generated fully captures the user's intention, no results are returned by DBpedia, hence the need for relaxation. The Precision@10 (again, this query returns more than 10 results) is equal to 1.0 which means that all returned results are of Grammy awarded guitarists.

8. RELATED WORK

Question answering (QA): Methods for natural-language QA (see[12, 26, 27, 32, 36, 43] and references given there) have traditionally cast questions into keyword queries in order to retrieve relevant passages from the underlying corpora and the Web. Then linguistic and statistical methods are used to extract answer candidates from passages, and to aggregate the evidence for candidate ranking and choosing the best answer(s). Some prior work (e.g., the START system [25]) made use of knowledge sources like the full text of Wikipedia but did not tap into structured data or knowledge. The IBM Watson system [23] recently achieved impressive improvements in QA, but made only limited use of structured data: 1) knowledge bases were used to answer question fragments that can be mapped into structured form with high confidence [10], and 2) checking lexical types of candidate answers in knowledge bases was used to prune spurious answers[10, 24].

Recently, research on QA over the Web of Linked Data has been pursued, and the QALD benchmark for evaluating progress has been defined [35]. In this setting, natural-language questions are mapped into structured queries, typically using the SPARQL language over RDF data [41]. The translation is performed either based on question/query templates [15, 40, 44] or by using algorithms for word-sense disambiguation [46]. The latter is most closely related to this paper. However, that prior work was limited to generating pure SPARQL queries. In our experiments, we will compare our approach to this baseline.

Keyword search: There is ample prior work on pure keyword queries over structured data, especially, relational databases (e.g., [1, 7, 18, 20, 47]). The semantics of such queries is to find matches in attributes of several relational records and then use foreign-key relationships to compute a connected result graph that explains how the various keyword matches relate to each other. One technique for this model is to compute so-called group-Steiner trees and use data statistics for ranking.

Telegraphic queries: None of the aforementioned keyword-search methods attempts to map the user input into struc-

Question	Generated Query	Relaxed	Ground truth answer size		Precision @10	
			QALD	Extended	QALD	Extended

"Swedish professional skateboarders"	`?x type Skateboarder .`
	`?x ?r1 Sweden .`
	`?x r2 ?y .`
	`?y type Professional .`
		`?x type Skateboarder {"professional"} .`
		`?x ?r1 Sweden .`
			2	3	0.2	0.3

"Which Greek goddesses dwelt on Mount Olympus?"	`?x type Greek_goddesses {"Olympus Mount dwelt"} .`	Not Needed	7	20	0.4	0.8
"guitarists awarded a" grammy"	`?x type guitarist .` `?x award Grammy_Award .`	`?x type guitarist {"awarded Grammy } Award"} .`	Pound et al. [34] telegraphic query workload		1.0	

Table 4: Examples of query generation and relaxation

tured predicates. Recent work on telegraphic queries [11, 33, 34, 37] pursues this very goal of translating such user requests into SQL or SPARQL queries. The focus here is on long keyword queries that describe entities or contain relational phrases that connect different entities and/or classes. The main challenge is to decompose the user input and infer an adequate query interpretation for often underspecified inputs such as "bands songs Stones". [34] uses a trained CRF for input decomposition and for mapping phrases onto semantic items; the training is based on query logs. [37] devised a probabilistic graphical model with latent variables for this purpose. Although the goal of these works is not really the same as ours, our experiments include the benchmark queries by [34] and a comparison to the performance of their technique.

Ranking: Statistical language models (LM's) [48] have been used for ranking the results of entity search (e.g., [4, 6, 17, 31, 42]). Also random-walk-based models have been developed for this purpose (e.g., [19, 38]). The typical setting here is to take a keyword query as input and return entities rather than documents, as evaluated in the TREC Entity track [5]. Alternatively, the user could enter a structured SPARQL query or a combination of SPARQL and keyword conditions, as currently pursued in the INEX Linked Data track [45]. LM-based ranking was taken one step further by [13, 14] who addressed the result ranking for full-fledged SPARQL queries (as opposed to mere entity search) with additional keyword conditions. We utilize this work and adapted these LM's to our setting.

9. CONCLUSION

With the explosion of structured data on the Web, translating natural-language questions into structured queries seems the most intuitive approach. However, state-of-the-art methods are not very robust as they tend to produce sophisticated queries that are correct translations yet return no answers from the underlying data. To enhance the robustness, we have proposed a new kind of optimization model followed by a suite of query relaxation techniques. Our experimental results with the QALD benchmark show that this approach can significantly enhance the quality of question answering.

10. REFERENCES

[1] Sanjay Agrawal, Surajit Chaudhuri, and Gautam Das DBXplorer: A System for Keyword-Based Search over Relational Databases. In ICDE, 2002.

[2] Nitish Aggarwal. Cross Lingual Semantic Search by Improving Semantic Similarity and Relatedness Measures. In ISWC, 2012.

[3] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary G. Ives. DBpedia: A Nucleus for a Web of Open Data. In ISWC, 2007.

[4] Krisztian Balog, Leif Azzopardi, and Maarten de Rijke. A language modeling framework for expert finding. *Inf. Process. Manage.* 45(1):1–19.

[5] Krisztian Balog, Leif Azzopardi, and Maarten de Rijke. Overview of the TREC 2011 Entity Track. In TREC, 2011.

[6] Krisztian Balog, Yi Fang, Maarten de Rijke, Pavel Serdyukov, and Luo Si. Expertise Retrieval. *Foundations and Trends in Information Retrieval* 6(2-3):127–256.

[7] Gaurav Bhalotia, Arvind Hulgeri, Charuta Nakhe, Soumen Chakrabarti, S. Sudarshan Keyword Searching and Browsing in Databases using BANKS. In ICDE, 2002.

[8] Stefan Büttcher, Charles L. A. Clarke, and Gordon V. Cormack. Information Retrieval: Implementing and Evaluating Search Engines. MIT Press, 2010.

[9] Elena Cabrio, Julien Cojan, Alessio Palmero Aprosio, Bernardo Magnini, Alberto Lavelli, and Fabien Gandon. QAKiS: an Open Domain QA System based on Relational Patterns. In ILD, 2012.

[10] Jennifer Chu-Carroll, James Fan, Branimir Boguraev, David Carmel, Dafna Sheinwald, and Chris Welty. Welty, C. Finding needles in the haystack: Search and candidate generation. *IBM Journal of Research and Development* 56(3):6.

[11] Marek Ciglan, Kjetil Nørvåg, and Ladislav Hluchý. The SemSets model for ad-hoc semantic list search. In WWW, 2012.

[12] Hoa Trang Dang, Diane Kelly, and Jimmy J. Lin. Overview of the TREC 2007 question answering track. In TREC, 2007.

[13] Shady Elbassuoni, Maya Ramanath, Ralf Schenkel, Marcin Sydow, and Gerhard Weikum. Language-model-based ranking for queries on rdf-graphs. In CIKM, 2009.

[14] Shady Elbassuoni, Maya Ramanath, and Gerhard Weikum. Query relaxation for entity-relationship search. In ESWC, 2011.

[15] Anette Frank, Hans-Ulrich Krieger, Feiyu Xu, Hans Uszkoreit, Berthold Crysmann, Brigitte Jörg, and Ulrich Schäfer. Question Answering from Structured Knowledge Sources. *Journal of Applied Logic* 5(1):20–48.

[16] Anthony Fader, Stephen Soderland, and Oren Etzioni. Identifying relations for open information extraction. In EMNLP, 2011.

[17] Hui Fang and ChengXiang Zhai. Probabilistic Models for Expert Finding. In ECIR, 2007.

[18] Vagelis Hristidis and Yannis Papakonstantinou. DISCOVER: Keyword Search in Relational Databases. In VLDB, 2002.

[19] Vagelis Hristidis, Heasoo Hwang, and Yannis Papakonstantinou. Authority-based keyword search in databases. *ACM Trans. Database Syst.* 33(1):1–40.

[20] Hao He, Haixun Wang, Jun Yang, and Philip S. Yu. BLINKS: ranked keyword searches on graphs. In SIGMOD, 2007.

[21] Heath, T., and Bizer, C. *Linked Data: Evolving the Web into a Global Data Space*. Morgan & Claypool Publishers, 2011.

[22] Johannes Hoffart, Fabian M. Suchanek, Klaus Berberich, and Gerhard Weikum. YAGO2: A spatially and temporally enhanced knowledge base from Wikipedia. *Artificial Intelligence* 194: 28-61.

[23] IBM 2012. Special Issue on "This is Watson" *IBM Journal of Research and Development* 56(3/4).

[24] Aditya Kalyanpur, J. William Murdock, James Fan, and Christopher A. Welty. Leveraging community-built knowledge for type coercion in question answering. In ISWC, 2011.

[25] Boris Katz, Sue Felshin, Gregory Marton, Federico Mora, Yuan Kui Shen, Gabriel Zaccak, Ammar Ammar, Eric Eisner, Asli Turgut, and L. Brown Westrick: CSAIL at TREC 2007 Question Answering. In TREC, 2007.

[26] Cody C. T. Kwok, Oren Etzioni, and Daniel S. Weld. Scaling Question Answering to the Web. In WWW, 2001.

[27] Lin, J. J.; An exploration of the principles underlying redundancy-based factoid question answering. *ACM Transactions on Information Systems* 25(2):1–48.

[28] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. Introduction to Information Retrieval. Cambridge University Press, 2008.

[29] Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. Generating typed dependency parses from phrase structure parses. In LREC, 2006.

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

[31] Zaiqing Nie, Yunxiao Ma, Shuming Shi, Ji-Rong Wen, and Wei-Ying Ma. Web object retrieval. In WWW, 2007.

[32] Anselmo Peñas, Eduard H. Hovy, Pamela Forner, Álvaro Rodrigo, Richard F. E. Sutcliffe, Caroline Sporleder, Corina Forascu, Yassine Benajiba, and Petya Osenova. Overview of QA4MRE at CLEF 2012: Question Answering for Machine Reading Evaluation. In *CLEF Evaluation Labs and Workshop*, 2012.

[33] Jeffrey Pound, Ihab F. Ilyas, and Grant E. Weddell. 2010. Expressive and Flexible Access to Web-extracted Data: A Keyword-based Structured Query Language. In SIGMOD, 2010.

[34] Jeffrey Pound, Alexander K. Hudek, Ihab F. Ilyas, and Grant E. Weddell. Interpreting keyword queries over web knowledge bases. In CIKM, 2012.

[35] Second Workshop on Question Answering over Linked Data (QALD-2). 2012. `http://www.sc.cit-ec.uni-bielefeld.de/qald-2`.

[36] Deepak Ravichandran and Eduard H. Hovy. Learning surface text patterns for a Question Answering System. In ACL, 2002.

[37] Uma Sawant and Soumen Chakrabarti Learning Joint Query Interpretation and Response Ranking. *CoRR* abs/1212.6193

[38] Pavel Serdyukov, Henning Rode, and Djoerd Hiemstra. Modeling expert finding as an absorbing random walk In SIGIR, 2008.

[39] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: a core of semantic knowledge. In WWW, 2007.

[40] Christina Unger, Lorenz Bühmann, Jens Lehmann, Axel-Cyrille Ngonga Ngomo, Daniel Gerber, and Philipp Cimiano. Template-based question answering over RDF data. In WWW, 2012.

[41] Christina Unger, Philipp Cimiano, Vanessa Lopez, Enrico Motta, Paul Buitelaar, and Richard Cyganiak. ILD 2012. `http://ceur-ws.org/Vol-913/`.

[42] David Vallet and Hugo Zaragoza. Inferring the most important types of a query: a semantic approach. In SIGIR, 2008.

[43] Ellen M. Voorhees. Overview of the TREC 2003 question answering track. In TREC, 2003.

[44] Sebastian Walter, Christina Unger, Philipp Cimiano, and Daniel Bär. Evaluation of a Layered Approach to Question Answering over Linked Data In ISWC, 2012.

[45] Qiuyue Wang, Jaap Kamps, Georgina Ramirez Camps, Maarten Marx, Anne Schuth, Martin Theobald, Sairam Gurajada, and Arunav Mishra. Overview of the INEX 2012 Linked Data Track. In *CLEF (Online Working Notes/Labs/Workshop)*, 2012.

[46] Mohamed Yahya, Klaus Berberich, Shady Elbassuoni, Maya Ramanath, Volker Tresp, and Gerhard Weikum. Natural Language Questions for the Web of Data. In EMNLP, 2012.

[47] Jeffrey Xu Yu, Lu Qin, and Lijun Chang *Keyword Search in Databases. Synthesis Lectures on Data Management*. Morgan & Claypool Publishers, 2010.

[48] ChengXiang Zhai. *Statistical Language Models for Information Retrieval. Synthesis Lectures on Data Management*. Morgan & Claypool Publishers, 2008.