Semantic Question Answering on Big Data

Marta Tatu, Steven Werner, Mithun Balakrishna, Tatiana Erekhinskaya, Dan Moldovan

Lymba Corporation
901 Waterfall Way, Bldg 5
Richardson, Texas 75080

{marta, steve, mithun, tatiana, moldovan}@lymba.com

ABSTRACT

This article describes a high-precision semantic question answering (SQA) engine for large datasets. We employ an RDF store to index the semantic information extracted from large document collections and a natural language to SPARQL conversion module to find desired information. In order to be able to find answers to complex questions in structured/unstructured data resources, our system produces rich semantic structures from the data resources and then transforms the extracted knowledge into an RDF representation. In order to facilitate easy access to the information stored in the RDF semantic index, our system accepts a user's natural language questions, translates them into SPARQL queries and returns a precise answer back to the user. Our improvements in performance over a regular free text search index-based question answering engine prove that SQA can benefit greatly from the addition and consumption of deep semantic information.

CCS Concepts

•Information systems → Data encoding and canonicalization; Query representation; Question answering;

Keywords

unstructured data; RDF; SPARQL; question answering

1. INTRODUCTION

The explosion of available knowledge makes the task of finding information by hand too expensive and complex. Aiming at returning brief answers from large data collections as responses to natural language questions, question answering (QA) systems are widely regarded as one of the next major contributions to the information technology world. Traditionally, they have been developed on top of free text search indexes as information retrieval (IR) systems enhanced with natural language processing (NLP) mechanisms with

varying processing depth. Despite these IR-based approaches to QA, there is no easy way to perform a federated search over both structured databases and unstructured text documents, including articles, manuals, reports, emails, blogs, and others. There is no easy way to enable more intelligent applications over such diverse data sources without considerable time and effort spent in system and data model customization by experts.

With the recent emergence of commercial grade Resource Description Framework (RDF) triple stores, it becomes possible to merge massive amounts of structured and unstructured data by defining a common ontology model for the DBMS schemas and representing the structured content as semantic triples. However, technology gaps exist. More specifically, there are no efficient and accurate algorithms and tools to transform unstructured document content into a rich and complete semantic representation that is compatible with the RDF standard. There are no methods for accessing information to enable intelligent applications while hiding the underlying complexity of the voluminous semantic data being searched.

In this paper, we describe new methods to transform unstructured data sources into a consolidated RDF store, merge it with other ontologies and structured data, and moreover, offer a natural language QA interface for easy use.

In a recent survey [2], Bouziane et al. divide QA systems (QAS) into: (1) QAS for the Web of documents and text that follow three main distinct subtasks: Question Analysis, Document Retrieval and Answer Extraction [9], and (2) QAS for the Web of data, which process questions in Natural Language, where they go through syntactic parsing with a step of named entity recognition and are classified according to predefined question categories. Finally, the SPARQL query is generated and executed against Linked Data.

The solution presented in this paper combines these two approaches: while the goal of our system is QA on a collection of documents, we use triple-based semantic representation and convert natural language questions into a SPARQL query.

The task of QA based on Linked Data has drawn attention of many researches in last ten years. Several QAS for Linked Data were reported in publications. For example, AquaLog [8], PowerAqua [7], NLP-Reduce [4] and FREyA [3]. Many of these systems map a natural language question to a triple-based representation. The open challenge on QA over Linked Data [16] has many participants. The competition has three tasks: multilingual question answering, biomedical question answering over interlinked data, and hybrid

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

SBD'16, July 01 2016, San Francisco, CA, USA

© 2016 ACM. ISBN 978-1-4503-4299-5/16/07...\$15.00

DOI: <http://dx.doi.org/10.1145/2928294.2928302>

question answering.

Most of the approaches use syntactic or semantic parsing to represent a question as a set of triples and then build a SPARQL query. The main differentiators include: (1) the linguistic processing tools used in the process and their coverage, (2) the knowledge bases used, and (3) the adaptability of the system to other knowledge bases.

Unger et al. [15] focused on the problem of generating SPARQL queries with aggregation and filter constructs. Such constructs are not captured using semantic triples as representation. The authors proposed a template-based approach to resolve the issue. SPARQL templates specify the query’s select clause, its filter and aggregation functions, as well as the number and form of its triples. The subject, predicate, and object of a triple are variables, some of which stand proxy for appropriate URIs. The main assumption of template generation is that the overall structure of the target SPARQL query is (at least partly) determined by the syntactic structure of the natural language question and by the occurring domain-independent expressions.

Yao and others focused on answering natural language questions using the Freebase knowledge base [18]. The authors contrast an AI approach and an IE approach to this problem and discover that the AI approach focuses on understanding the intent of the question, via shallow or deep forms of semantic parsing (e.g., the lambda calculus), and then mapping to database queries. Performance is thus bounded by the accuracy of the original semantic parsing, and the well-formedness of resultant database queries. The IE community, where the authors belong, approaches QA differently: first performing relatively coarse information retrieval as a way to triage the set of possible answer candidates, and only then attempting to perform deeper analysis. The authors use inference to match Freebase relation types with words in the question (e.g., *brother* and *male sibling*). The authors show that relatively modest IE techniques, when paired with a webscale corpus, can outperform these sophisticated approaches by roughly 34% relative gain.

Recently, joint approaches were applied to the task. Yahya [17] used 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.

CASIA system [6] uses Markov Logic Networks (MLNs) for learning a joint model for phrase detection, phrases mapping to semantic items and semantic items grouping into triples. Markov logic is used to describe conflict resolution rules for disambiguation. Hard clauses represent mutual exclusivity constraints between items and relations. The soft constraints describe associations between part-of-speech tags of phrases and types of mapped semantic items, between the dependency tags in the dependency pattern path of two phrases and the types of relations of two mapped semantic items, and influence of other features on the disambiguation decision. The authors used DBpedia as their knowledge base.

Another Markov logic-based approach [5] investigates three ways of applying MLNs to the QA task. First, the extraction of science rules directly as MLN clauses and exploiting the structure present in hard constraints to improve tractability. Second, interpreting the science rules as describing prototypical entities, resulting in a drastically simplified but brittle

Figure 1: System Overview

network. The third approach uses MLNs to align lexical elements as well as define and control how inference should be performed in this task.

One more relevant research direction is natural language interface for data bases [13]. Given a question, PREISE system determines whether it’s tractable and if so, outputs the corresponding SQL query. The problem of mapping from a complete tokenization to a set of database elements is reduced to a graph matching problem, which is solved with a max-flow algorithm.

2. TRIPLE-BASED QA

Our QA engine leverages an existing natural language processing (NLP) module, which is employed to pinpoint the semantics of both the document collection as well as the user’s input question. In the document indexing phase, the semantic information derived from document content is represented in an RDF format that facilitates its storage into an RDF store. At query time, the natural language question of the user is parsed to identify its meaning, which is then automatically converted into a SPARQL query that will be used to query the already populated RDF store (Figure 1).

The knowledge extraction module and the question processing tool must share the NLP engine to ensure that the semantics extracted from both the document collection as well as the input question are similar and can be represented using the same set of triples.

Since the goal of this article is to detail the means of answering natural language questions using an RDF-triple-based framework, we will omit the description of the NLP module, which can be any suite of NLP tools that identify named entities, word senses, coreferencing entities, and semantic relationships between the concepts of a document content. A more detailed description of one such knowledge extraction engine can be found in [14].

2.1 RDF Store

Having extracted various semantic knowledge from input documents, and, therefore, having created a more structured dataset from the unstructured input text, we define a robust RDF representation, which when translated into n-triples, can be stored within an RDF semantic index. This store can then be accessed, visualized, queried, or integrated with already available structured data. For this purpose, we use Oracle 12c, which can efficiently index and access large amounts of RDF triples [12].

We note that, in addition to the RDF representation of the input document collection (described below in Section 2.1.1), the RDF store may contain n-triples that define a domain ontology or even WordNet. For these knowledge resources, we store any concept information that they provide (part-of-speech, sense number, named entity class, if available)

as well as any semantic relationships identified between the ontological concepts.

2.1.1 RDF Representation

The available knowledge extracted from the document content includes: (1) lexical knowledge (sentence boundaries, token information, including part-of-speech tag, start and end positions, and lemma), (2) syntactic information (head-of-parse-phrase flags for tokens, and syntactic phrase dependencies), as well as (3) semantic knowledge (named entity labels, WordNet word senses, coreference chains, and semantic relations). Storing all this information in an RDF store creates a large number of RDF triples, which proves to be intractable for large collections of textual data (for instance, 6.3MB of unstructured plain text can produce 13,314,886 n-triples or 1.03GB of RDF XML). However, not all the available linguistic knowledge is needed for a reliable QA system.

The semantic information is the most valuable to an end consumer of the knowledge conveyed by the input text. Therefore, the subset of linguistic knowledge translated to RDF includes: (1) only tokens that participate in semantic relationships; (2) the semantic relations linking these tokens. For named entity tokens, we store only their entity type, lemma, synset information (if available), and reference sentence. For all other tokens, we keep only their surface form, lemma, synset information, a boolean is-event flag, and reference sentence. Each semantic relation is described by its arguments and its type. These include any coreference links identified during NLP. This reduced representation, which is semantically equivalent to the schema that represents all available linguistic knowledge, reduces the size of the RDF store by 50%.

This sample RDF XML file portion describes tokens *Russia* and *won* as well as their semantic link `AGENT(Russia, won)`:

```
<rdf:Description rdf:about="http://www.lymba.com/rdf/7a1c9e82/Token/0_5">
  <rdf:type rdf:resource="http://www.lymba.com/rdf/terms/v1.0/Token"/>
  <lymterms:inSentence rdf:resource="http://www.lymba.com/rdf/7a1c9e82/S/0_53"/>
  <lymterms:text>Russia</lymterms:text>
  <lymterms:hasCountryName>Russia</lymterms:hasCountryName>
  <lymterms:synsetId>108444912</lymterms:synsetId>
</rdf:Description>
<rdf:Description rdf:about="http://www.lymba.com/rdf/7a1c9e82/Token/7_9">
  <rdf:type rdf:resource="http://www.lymba.com/rdf/terms/v1.0/Token"/>
  <lymterms:inSentence rdf:resource="http://www.lymba.com/rdf/7a1c9e82/S/0_53"/>
  <lymterms:text>won</lymterms:text>
  <lymterms:lemma>win</lymterms:lemma>
  <lymterms:isEvent>true</lymterms:isEvent>
  <lymterms:synsetId>201065094</lymterms:synsetId>
</rdf:Description>
<rdf:Description rdf:about="http://www.lymba.com/rdf/7a1c9e82/SR/0_5-isAgentOf-7_9">
  <rdf:type rdf:resource="http://www.lymba.com/rdf/terms/v1.0/SR"/>
  <lymterms:type>isAgentOf</lymterms:type>
  <lymterms:confidence>0.856003</lymterms:confidence>
  <lymterms:lhs rdf:resource="http://www.lymba.com/rdf/7a1c9e82/Token/0_5"/>
  <lymterms:rhs rdf:resource="http://www.lymba.com/rdf/7a1c9e82/Token/7_9"/>
</rdf:Description>
```

2.1.2 Reasoning on the RDF Store

Our deep semantic QA engine uses various types of RDF entailment rules to augment the semantic information extracted from textual content. Inferences are necessary for answering any non-trivial questions by deriving implicit relationships and making the data smarter. They complete the knowledge extracted from textual content and lead to shorter queries and fast response times. The entailment rules are preloaded and the reasoning process is performed as soon as the n-triples capturing the documents' meaning are indexed in the RDF store.

The first variety of RDF rules involves the generation of *sameAs* predicates as coreferring entities within- as well as across-documents (e.g., all mentions of *Russia* refer to the

Figure 2: RDF Store

same entity). Once these new n-triples are created, an OWL-PRIME entailment will cluster all URIs of coreferring tokens, thus, gathering their corresponding semantic properties to a single URI.

Another type of RDF-based axioms combine two semantic relations that share an argument. For instance, $\text{LOCATION}(x, y) \& \text{PART-WHOLE}(x, z) \rightarrow \text{LOCATION}(z, y)$ can be defined as an RDF entailment rule $(?tok \text{ lymterms:isLocationOf } ?lhs) (?rhs \text{ lymterms:isPartOf } ?tok) \rightarrow (?rhs \text{ lymterms:isLocationOf } ?lhs)$ to generate new LOCATION instances from existing LOCATION and PART-WHOLE relations. We note that these combinations of semantic relations can be applied to semantic links identified within the document content as well as semantic relations defined within domain ontologies or WordNet between concepts. Other examples include the transitivity of HYPERNYMY, PART-WHOLE, or KINSHIP relationships. For a more formal description of these axioms as well as methodologies on how to generate them, please refer to [1].

Lexical chains¹ of WordNet relations can be translated to RDF entailment rules that generate new n-triples between semantically-linked concepts. For instance, $\text{victim:n\#1} \leftarrow \text{IS-A} \leftarrow \text{martyr:n\#1} \rightarrow \text{DERIVATION} \rightarrow \text{martyr:v\#1} \rightarrow \text{IS-A} \rightarrow \text{kill:v\#1}$ links *victim* and *kill* and enables the extraction of *It was reported that a Gulf Cartel member killed a top Zeta lieutenant named Victor Mendoza.* as an answer for *Which Zeta lieutenant was a victim of the Gulf Cartel?*

Last but not least, entailment rules can be defined to create *inSynset* n-triples between a document token and its corresponding ontological concept/WordNet synset.

We note that this new knowledge is seamlessly blended into the RDF semantic index and is made available at query time. The RDF reasoning step that employs all these types of axioms to complete the knowledge explicitly conveyed by document content finalizes the collection indexing phase of our deep semantic QA process.

2.2 Natural Language to SPARQL

Given our RDF-based QA framework, the question answering phase of the QA process must generate SPARQL

¹A lexical chain is defined as a weighted path between two synsets, where the path consists of a sequence of one or more synsets connected by relations in WordNet.

queries semantically equivalent to input questions and use these queries to interrogate the RDF store. In order to ensure the system’s robustness, we implemented several query relaxation procedures to be used when no answers are found in the RDF store. If competing answers are identified, a ranking module is used to create the system’s final response.

2.2.1 Question Processing

The first step in this process is to derive the semantic representation of the input question. In addition to processing the question text using the deep NLP engine employed for the document content, more information about the question’s expected answer must be derived (e.g., *_human* or *_organization* for a who-question, *_date* or *_time* for a when-question, etc.) as well as its answer type term(s) (e.g., *cartel* for *which cartel*-questions). We are using a hybrid system, which takes advantage of precise heuristics as well as machine learning algorithms for ambiguous questions. A maximum entropy model was trained to detect both answer type terms and answer types. The learner’s features for answer type terms include part-of-speech, lemma, head information, parse path to WH-word, and named entity information. Answer type detection uses a variety of attributes such as additional answer type term features and set-to-set WordNet lexical chains, which link the set of question keywords to the set of potential answer type nodes [10, 11].

2.2.2 SPARQL Query Formulation

SPARQL queries are not easy to write even when knowledgeable about the content of the RDF store, especially for complex queries. Therefore, we developed a natural language (NL) to SPARQL conversion module that generates SPARQL queries equivalent to a question’s semantic representation.

For a given question, the answer type and answer type term information is used to decide which SPARQL variables are to be SELECTED and returned by the query. Furthermore, the set of semantic relations identified within the question text describe the SPARQL query’s WHERE constraints – the triple patterns that must be satisfied by the retrieved answers (Figure 3).

More specifically, a unique SPARQL variable name is associated with each question token that participates in a semantic relation and is linked to the answer variable through a chain of semantic relationships. The answer variable is included in the SELECT clause of the SPARQL query along with a variable that denotes the text of the sentence which contains that answer (e.g., `SELECT ?ans ?sentenceText`).

The WHERE clause of the SPARQL query contains the semantic triples that must be satisfied by the indexed knowledge. First, the answer’s semantic constraints are added:

1. answer type filtering (e.g., for an *_organization* answer type: `?answer lyterms:hasOrganizationName ?ans` – which forces the answer to be an organization named entity; if no answer type is identified, the query contains `?answer wn:hasLexeme ?ans`),
2. answer type term tests (e.g., for the *cartel* answer type term: `?answer rdf:subClassOf wn:synset-cartel-noun-1 . ?answer wn:hasLexeme ?ans` – which constrains the answer to hyponyms of the answer type term *cartel*), and

Figure 3: NL to SPARQL

3. sentence information (`?answer lyterms:sentence ?sentenceText`).

The semantic relation constraints follow. For each semantic relation, one to three triple patterns are created. The first is mandatory and describes the relation (e.g. `?answer lyterms:isAgentOf ?trade` for `AGENT(who,trade)`; `?gun lyterms:isThemeOf ?trade` for `THEME(gun,trade)`; `?illegally lyterms:isMannerOf ?trade` for `MANNER(illegally,trade)`). The other two describe the variables involved if not described already (e.g., `?trade lyterms:inSynset wn:synset-trade-verb-1`).

Using this SPARQL generation procedures, the question *Which cartels trade guns illegally?* is converted into

```
SELECT ?ans ?sentenceText
WHERE {
  ?answer rdf:subClassOf wn:synset-cartel-noun-1 .
  ?answer wn:hasLexeme ?ans .
  ?answer lyterms:sentence ?sentenceText .
  ?answer lyterms:isAgentOf ?trade .
  ?trade lyterms:inSynset wn:synset-trade-verb-1 .
  ?gun lyterms:isThemeOf ?trade .
  ?gun lyterms:inSynset wn:synset-gun-noun-1 .
  ?illegally lyterms:isMannerOf ?trade .
  ?illegally lyterms:inSynset wn:synset-illegally-adverb-1 .
}
```

2.2.3 Query Relaxation

We note that the query generated using the information extracted from the user’s question includes *inSynset* links to concepts chosen by the user. Since, these may not be identical to the correct answer’s concepts as indexed in the RDF store, if no rows are selected for this initial SPARQL query, we relax the query to allow for hyponyms, parts, derivations as well as concepts linked by lexical chains to the question terms to be matched by the RDF reasoning engine (e.g., `?trade lyterms:inSynset wn:synset-trade-verb-1` is replaced with `?trade inSynset ?syn . ?syn wn:hyponymOf wn:synset-trade-verb-1`, which allows `?trade` to be matched by any hyponym of *trade* such as *traffic*, *import*, *export*, etc.).

Furthermore, if no answers are identified using these relaxed queries, further relaxations to the set of WHERE con-

straints are made: variable-describing triples and semantic-relation-definition triples are dropped, starting with the variables that have the smallest impact on the answer variable. For the example shown above, these include `?illegal` and `?gun` (in this order); `?trade` is semantically linked to the desired answer and will not be relaxed.

Future research must be conducted to identify the optimum relaxation path, e.g., the optimal order in which query terms are to be replaced by their hyponyms or meronyms as well as the best order in which semantic relation triples are to be dropped from the automatically generated SPARQL query.

Furthermore, we note that our NL to SPARQL procedure can be also employed for an existing RDF store, where the indexed document collection is not represented using the same set of semantic relations as the one identified within the input question. For all such cases, the question’s semantic graph (as returned by the NLP tools) must be mapped to the documents’ set of n-triple predicate, prior to its conversion to a SPARQL graph. For this purpose, semantic rules must be defined that combine the relations identified within the question to pinpoint the types of relations stored for the document content. For instance, if the RDF store contains `ARRESTED_AT` n-triples, which cannot be extracted natively by the NLP tools from the question input, a semantic rule similar to `THEME(x,y) & LOCATION(l,y) & ISA(y,arrest) → ARRESTED_AT(x,l)` would bridge a question’s semantics to the RDF store predicates. The new question semantic graph, which makes use of the store’s predicates, can be accurately converted into the correct SPARQL query.

3. EXPERIMENTS AND RESULTS

3.1 Experimental Data

For the development and evaluation of our SQA system, we used a collection of documents about the Illicit Drugs domain: 584 documents, including Wikipedia articles about drugs as well as web documents regarding illicit drug trade, production, usage, and related violence, including cartel information for use in law enforcement. The size of this collection is (1) 6.3MB of plain unstructured text, (2) 650MB of NLPXML – a structured format used to encode the rich body of semantic information derived from the documents’ content, and (3) 546MB of RDF XML for a total of 6,729,854 RDF triples.

We also made use of a custom ontology generated for this domain, which includes 13,901 concepts and 33,359 semantic relations between these concepts. Its RDF representation consists of 232,585 n-triples.

The system was applied for monitoring treatment efficiency reported in biomedical papers from PUBMED. Currently, it extracts full semantic representation and provides real-time QA for the collection of 7 million documents. No formal quality evaluation is currently available for this collection.

3.2 Results

The mean reciprocal rank (MRR) is the common metric used for QA systems that produce a list of possible responses ordered by probability of correctness. Used to score systems participating in the NIST TREC evaluations, it is the average of the reciprocal ranks of results for a question set. More specifically, for a set of questions Q ,

$$MRR = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{1}{rank_i},$$

where $rank_i$ is the rank of the first correct answer for question i .

3.2.1 Question Answering

For the 344 questions we created for the Illicit Drugs collection, we measured a 65.82% MRR score for our SQA engine. This is a 19.31% increase in performance when compared to the 46.51% MRR achieved using a free text-search index alone. This baseline result was obtained by using PowerAnswer [10, 11].

The set of questions used to evaluate the SQA system vary in complexity. In addition to factoid questions (e.g., *Who are the leaders of the Juarez Cartel?*), which make up 49% of the question test set, we used definition questions (e.g., *What are tweezers?*) – about 34% of the test set, and yes/no questions (e.g., *Is the Cartel of Los Zetas active in the US?*), list questions (e.g., *What are some effects of using Methamphetamine?*), and few procedural questions (e.g., *How is cocaine manufactured?*).

Our system performed well on factoid questions (85.46% MRR), definition questions (78.19%) and list questions (68.02%). It found one or more answers for only 87.2% of the 344 questions despite our query relaxation procedures.

3.2.2 NL to SPARQL

In order to evaluate the text-to-SPARQL conversion module, we manually converted into SPARQL queries 34 questions (10%) randomly selected from our test set. While creating these gold annotation queries, we followed the same guidelines with respect to SPARQL variable naming (e.g., use the concept’s lemma as its corresponding variable name, if not already assigned to another concept) and SPARQL triple pattern ordering in order to make the comparisons to the queries automatically generated by our SQA engine easier.

Given the transformation procedure described in 2.2.2, the `SELECT` clauses of SQA queries are identical to their corresponding clauses of gold queries for 85.29% questions. The `SELECT` clause is rather simple and only in few cases it requires additional variable names.

The accuracy of the `WHERE` clause generation process is 67.64% at the question level (an automatically generated `WHERE` clause is marked as correct if it matches the gold annotation in its entirety, i.e., all triple patterns are correct; order is not important). At the triple pattern level, the `WHERE` clause generation module has a precision of 78.20%, creating correctly 122 triple patterns out of the 156 produced by the human conversion of the questions into SPARQL queries. We note that this evaluation process was performed for the SPARQL queries generated directly from the natural language question with no relaxation of the queries.

The SPARQL query relaxation module was employed during the processing of 68.54% of the questions. It is vital to the robustness of our SQA system. The relaxation of initial query terms only (*inSynset* relaxation) was sufficient for 30.98% of the test SPARQL queries. For remaining queries, one or more semantic relation triple pattern was dropped from the SPARQL query in an effort to identify an answer within the RDF store.

3.2.3 Error Analysis

The system relies heavily on the semantic relations identified within the content of analyzed documents as well as relationships identified within input questions. Semantic parsing is a high level NLP processing step affected by errors propagated from other NLP tools. For instance, when processing the question *What over-the-counter drug is used as a recreational drug?*, if the collocation identification module fails to recognize *over-the-counter* as a multi-word adjective, the syntactic parser creates a prepositional phrase with *over* as its syntactic head and an incorrect *LOCATION(counter, drug)* semantic relation is derived. The answer type term (*drug*) is one of its argument. Therefore, this semantic relation will not be dropped during the SPARQL query relaxation process and no answers are returned from the SQA system.

72.7% of the errors made by our SQA engine were caused by faulty or missing semantic relations within the answer passage and/or the input question. These relationships also affect the quality of the semantic relations derived using the semantic closure axioms. They also influence the correctness of the SPARQL queries automatically generated from natural language questions.

16.3% of the errors were caused by the NL-to-SPARQL conversion module. New strategies must be implemented for an accurate transformation of yes/no questions and procedural questions. Furthermore, this conversion process depends heavily on the correctness of the question processing module, which determines the question's answer type and answer type term. The errors caused by SPARQL queries that are incorrect due to inaccurate or missing semantic relations were included in the previous category of system mistakes.

4. CONCLUSIONS

In this paper, we describe a triple-based semantic question answering system that stores the rich semantic structured identified in unstructured data sources into scalable RDF stores and facilitates easy access to the stored knowledge by allowing its users to ask natural language questions that are automatically converted into SPARQL queries and used to interrogate the RDF semantic index. This semantic approach to question answering yields more accurate results when compared with a free-text search index-based question answering engine.

5. REFERENCES

- [1] E. Blanco and D. I. Moldovan. Unsupervised learning of semantic relation composition. In *Proceedings of HLT-2011*, pages 1456–1465, 2011.
- [2] A. Bouziane, D. Bouchiha, N. Doumi, and M. Malki. Question Answering Systems: Survey and Trends. *Procedia Computer Science*, 73:366 – 375, 2015. AWICT 2015.
- [3] D. Damjanovic, M. Agatonovic, and H. Cunningham. FREyA: An Interactive Way of Querying Linked Data Using Natural Language. In *Proceedings of ESWC'11*, pages 125–138, 2012.
- [4] E. Kaufmann. *Talking to the Semantic Web ? Natural Language Query Interfaces for Casual End-users*. PhD thesis, University of Zurich, Feb. 2009.
- [5] T. Khot, N. Balasubramanian, E. Gribkoff, A. Sabharwal, P. Clark, and O. Etzioni. Markov Logic Networks for Natural Language Question Answering. *CoRR*, abs/1507.03045, 2015.
- [6] K. Liu, J. Zhao, S. He, and Y. Zhang. Question Answering over Knowledge Bases. *IEEE Intelligent Systems*, 30(5):26–35, 2015.
- [7] V. Lopez, M. Fernández, E. Motta, and N. Stieler. PowerAqua: Supporting users in querying and exploring the Semantic Web. *Semantic Web*, 3(3):249–265, 2012.
- [8] V. Lopez, V. Uren, E. Motta, and M. Pasin. AquaLog: An ontology-driven question answering system for organizational semantic intranets. *Web Semantics: Science, Services and Agents on the World Wide Web*, 5(2), 2007.
- [9] V. Lopez, V. Uren, M. Sabou, and E. Motta. Is Question Answering Fit for the Semantic Web?: A Survey. *Semantic Web*, 2(2):125–155, Apr. 2011.
- [10] D. Moldovan, M. Bowden, and M. Tatu. A Temporally-Enhanced PowerAnswer in TREC 2006. In *Proceedings of TREC 2006*, 2006.
- [11] D. Moldovan, C. Clark, and M. Bowden. Lymba's PowerAnswer 4 in TREC 2007. In *Proceedings of TREC 2007*, 2007.
- [12] Oracle. RDF Semantic Graph Prerequisites, and Advanced Performance and Scalability for Semantic Web Applications, 2014.
- [13] A.-M. Popescu, O. Etzioni, and H. Kautz. Towards a Theory of Natural Language Interfaces to Databases. In *Proceedings of UII '03*, pages 149–157, 2003.
- [14] M. Tatu, T. E. Mithun Balakrishna, Steven Werner, and D. Moldovan. Automatic Extraction of Actionable Knowledge. 2016.
- [15] C. Unger, L. Bühmann, J. Lehmann, A.-C. Ngonga Ngomo, D. Gerber, and P. Cimiano. Template-based Question Answering over RDF Data. In *Proceedings of WWW '12*, pages 639–648, 2012.
- [16] C. Unger, C. Forascu, V. Lopez, A.-C. N. Ngomo, E. Cabrio, P. Cimiano, and S. Walter. Question Answering over Linked Data (QALD-5). In L. Cappellato, N. Ferro, G. J. F. Jones, and E. SanJuan, editors, *CLEF (Working Notes)*, volume 1391 of *CEUR Workshop Proceedings*, 2015.
- [17] M. Yahya, K. Berberich, S. Elbassuoni, M. Ramanath, V. Tresp, and G. Weikum. Natural Language Questions for the Web of Data. In *Proceedings of EMNLP-CoNLL '12*, pages 379–390, 2012.
- [18] X. Yao, J. Berant, and B. Van Durme. Freebase QA: Information Extraction or Semantic Parsing? In *Proceedings of ACL Workshop on Semantic Parsing*, 2014.