Open Access

Open Journal of Semantic Web (OJSW)
Volume 3, Issue 1, 2016

http://www.ronpub.com/ojsw
ISSN 2199-336X

A Semantic Question Answering Framework for Large Data Sets

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

Lymba Corporation, 901 Waterfall Way, Bldg 5, Richardson, Texas 75080, USA,
{marta, mithun, steve, tatiana, moldovan}@lymba.com

Abstract

Traditionally, the task of answering natural language questions has involved a keyword-based document retrieval step, followed by in-depth processing of candidate answer documents and paragraphs. This post-processing uses semantics to various degrees. In this article, we describe a purely semantic question answering (QA) framework for large document collections. Our high-precision approach transforms the semantic knowledge extracted from natural language texts into a language-agnostic RDF representation and indexes it into a scalable triplestore. In order to facilitate easy access to the information stored in the RDF semantic index, a user's natural language questions are translated into SPARQL queries that return precise answers back to the user. The robustness of this framework is ensured by the natural language reasoning performed on the RDF store, by the query relaxation procedures, and the answer ranking techniques. The improvements in performance over a regular free text search index-based question answering engine prove that QA systems can benefit greatly from the addition and consumption of deep semantic information.

Type of Paper and Keywords

Regular research paper: *RDF, SPARQL, unstructured data, large datasets, question answering (QA), reasoning, ontology*

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, robust and precise 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) [22] 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 (1) transform unstructured data into a structured format, (2) merge it with other ontologies and structured data into a consolidated RDF store, and (3) offer a natural language QA interface for easy use. To make the QA robust, we propose various natural language reasoning methodologies, query relaxation procedures, and an answer ranking technique.

This novel framework operates on the semantics of the natural language text and is, therefore, language-independent. It is also highly flexible, allowing a system designer to tailor its various components to a particular task and/or document collection as well as giving them the liberty to choose from the many available triplestores.

The specific implementation described in this article was developed on top of Oracle 12c Spatial and Graph [19] and makes use of Lymba's knowledge extraction capabilities [24]. We evaluated and measured its performance on a variety of questions about illicit drugs. However, we show the impact of the QA framework's various modules on three different data sets. And, although parts of the framework detailed below have been introduced in our *Semantic Question Answering on Big Data* article [25], the customization of the framework for different domains or needs is novel along with the description of our answer ranking module and the many examples and details added to various sections of the original article.

1.1 Related Work

In a recent survey [4], Bouziane et al. divide QA systems into:

1. *QA for web of documents and text* that follow three main distinct subtasks: Question Analysis, Document Retrieval, and Answer Extraction [16] to process natural language questions and retrieve precise answers from textual documents.

2. *QA for web of data* that apply Named Entity Recognition, Syntactic Parsing, Question Classification, and SPARQL Generation on natural language questions and retrieve precise answers from Linked Data.

Development of *QA for web of documents and text* has been the center of research in the IR and NLP communities for several decades. Such QA systems have been developed, hardened, and evaluated under several government funded programs including National Institute of Standards and Technology (NIST) TREC QA competition from 1999 until 2007 [29, 6]. These QA systems rely on shallow, named entity based indexing to retrieve a small set of candidate answer documents from large collections. The candidate answer documents undergo deep semantic analysis in a post-processing phase to retrieve the final answers. The TREC QA competitions have revealed that the systems perform very well on processing and retrieving precise answers for factoid questions that mainly query for date, time, location, person, and organization named entities but do not perform well on list and definition questions [18, 17]. The lack of semantic knowledge being indexed and queried in the document retrieval phase results in low coverage of answer candidate sentences/documents for further analysis and processing, and thus leads to non-optimal performance on certain types of questions.

QA for web of data has lately drawn the attention of many researchers and has resulted in the development of several QA systems for Linked Data such as Aqualog [15], PowerAqua [14], NLP-Reduce [11] and FREyA [5]. The advent of several competitions in this arena including the Open Challenge on Question Answering over Linked Data [28] has helped in data sharing and development of robust systems. Most approaches to *QA for web of data* use dependency or syntactic parsing to extract and represent a question's semantics as a set of triples and then build a SPARQL query. The solutions differentiate themselves mainly on: (1) the linguistic processing tools and their performance, (2) usage of knowledge bases, and (3) the ease of adaptation to newer domains.

Unger et al. [27] focused on applying aggregation and filter constructs to resolve SPARQL query generation related issues. The authors proposed a template-based approach to SPARQL query generation and handle constructs that are not captured using semantic triple representation. The SPARQL templates specify the query's select clause, its filter and aggregation functions, as well as the number and form of the semantic triples. The semantic triples are represented appropriately by variables and data elements from the natural language question. 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 et al. [34] focused on answering natural language questions using the Freebase knowledge base. Their solution compares an artificial intelligence (AI) approach against an information extraction (IE) approach. The

AI based 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 extracted semantics to database queries. The AI module's performance is thus bounded by the accuracy of the original semantic parsing and the well-formedness of resultant database queries. The IE based approach first performs relatively coarse information retrieval as a way to triage the set of possible answer candidates, and then attempts to perform deeper analysis to retrieve the correct answer. The deeper analysis includes use of inference to match Freebase relation types with words in the question (e.g., *brother* and *male sibling*) and answer candidates. The authors show that relatively modest IE techniques when paired with a webscale corpus can outperform sophisticated approaches by roughly 34% relative gain.

Recently, joint approaches have been applied to solve several related and dependent tasks in QA. Yahya [33] used an integer linear program to jointly solve: 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 [13] uses Markov Logic Networks (MLNs) [23] for learning a joint model for phrase detection, mapping phrases to semantic items, and grouping semantic items into triples. Markov logic clauses are used to describe conflict resolution rules for disambiguation. Hard clauses represent mutual exclusivity constraints between items and the relations that they partake in. The soft constraints describe: 1) association between phrases and semantic items; 2) association between dependency tags in the dependency pattern path of two phrases and the relationship types between the phrases' mapped semantic items; 3) influence of other features on the disambiguation decision. The CASIA system uses DBpedia [7] as its knowledge base.

Another Markov logic-based approach [12] investigates three ways of applying MLNs to the QA task. First, the approach extracts science rules directly as MLN clauses and exploits the structure present in hard constraints to improve tractability. Second, it interprets the science rules to describe prototypical entities, and this results in a drastically simplified but brittle network. Lastly, the 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 related to QA is the creation of natural language interfaces for querying structured information loaded in databases [20]. Given a natural language question, the PRESISE system [20] first determines if the question is relevant to the structured information loaded in a database and whether a tractable SQL can be generated. The database is then queried using the automatically generated SQL to retrieve answers. The problem of mapping the natural language question represented by a set of tokenized concepts to a set of database elements is reduced to a graph matching problem, which is solved with a max-flow algorithm.

The framework presented in this paper bridges QA approaches for a web of textual documents and structured data content. We describe the means to represent the knowledge extracted from textual documents and structured data as RDF triples, and then convert users' natural language questions into SPARQL [31] for querying and retrieving answers from the RDF store. The goal of the described framework is to perform high precision QA on large heterogenous data sets containing both structured and unstructured knowledge.

2 TRIPLE-BASED QUESTION ANSWERING

Our semantic question answering (SQA) framework leverages an existing natural language processing (NLP) module. This suite of tools pinpoints 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 a semantic triple store. At query time, the user's natural language question is parsed to identify its meaning. This is then automatically converted into a SPARQL query that will be used to retrieve precise answers from the already populated RDF store (Figure 1). All methodologies described in this article are language agnostic.

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 identifies named entities, word senses, coreferring entities, and semantic relationships between concepts from the content of a document. A detailed description of one such knowledge extraction engine can be found in [24]. We emphasize the required degree of semantic processing (deep NLP), which goes beyond identification of named entity types and/or word senses alone (shallow NLP). All these highly semantic components are required to capture the meaning of the input, to create a faithful semantic representation of the input documents and questions, and to produce accurate

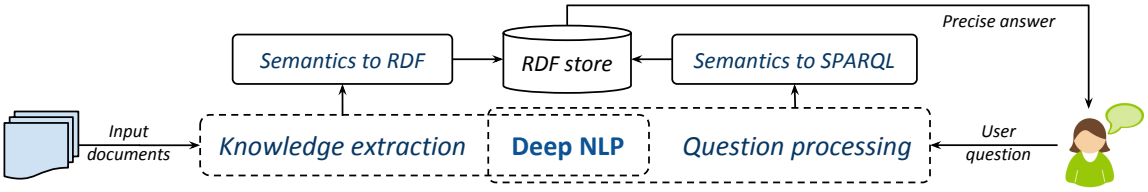

Figure 1: Overview of the semantic question answering framework

answers for input questions.

2.1 RDF Store

In this section, we detail the novel steps of the document indexing phase of our proposed SQA framework (the complete step-by-step process shown below). More specifically, we present the RDF representation of the input document content as well as the various types of entailment rules that can be used on an RDF store to generate additional triples.

```
Input: Document collection
Output: RDF semantic index

1 for each input document:
  1.1 Deep NLP of document content
  1.2 RDF representation of its semantics
2 load document RDF triples in RDF store
3 load custom/domain ontology/WordNet in
triplestore
4 define entailment rules for reasoning
on the RDF store
5 generate new triples using already
defined entailment rules
```

Having extracted various semantic knowledge from the input documents (step 1.1), and, therefore, having created a more structured dataset from the unstructured input text, we define a robust RDF representation (step 1.2), which when translated into triples, can be stored within an RDF semantic index (step 2). This store can then be accessed, visualized, queried, or integrated with already available structured data. For this purpose, any scalable triplestore may be used. For the implementation evaluated in this paper, we use Oracle 12c Spatial and Graph, which can efficiently index and access large amounts of RDF triples [19].

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 triples that define a domain ontology or even WordNet [9, 32] (step 3)[1]. 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. All RDF XML files are translated to triples using Apache Jena [26].

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). However, 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 triples or 1.03GB of RDF XML). Fortunately, not all this 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, we reduce the set of linguistic knowledge translated to RDF to include: (1) only concepts that participate in semantic relationships, and (2) the semantic relations linking these concepts. More specifically, for named entities, we store only their entity type, lemma, synset information (if available), and reference sentence. For temporal and locative expressions, we make use of their normalized values when setting the concept's properties (e.g., *2008-05-20* is the lemma for both *May 20th of 2008* and *May 20, 2008*). For all other concepts, we keep only their surface form, lemma, synset information, a boolean is-event flag, and reference sentence. Each semantic relation is represented as an additional triple for its left-hand-side argument URI, which points to its right-hand-side argument URI. These include any coreference links identified during the NLP process. This reduced representation, which is *semantically* equivalent to the schema that represents all available linguistic knowledge, reduces the size of the RDF store

[1] The loading of ontological triples into the RDF store is a one-time process. The triples can be shared by different semantic models.

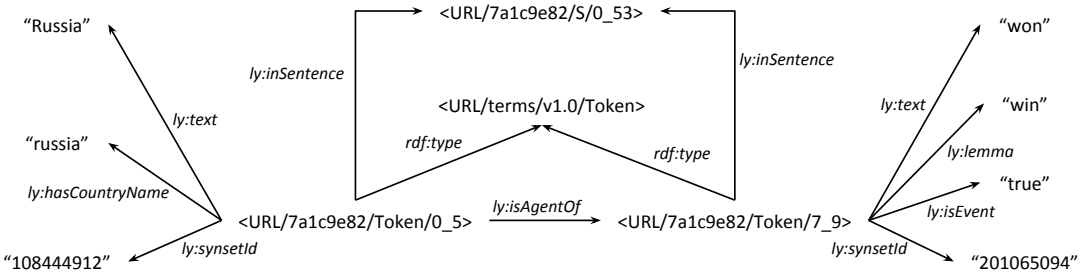

Figure 2: Sample RDF graph generated using the RDF representation proposed in this article

by 50%, while still maintaining the source sentences and the document of origin for a concept or relation. However, if additional information is required for the presentation of the answer to the user (e.g., highlight the exact answer of a question within its sentence), this RDF representation can be augmented accordingly with non-semantic information (e.g., include start/end offset properties for all concepts).

The following sample RDF XML describes tokens *Russia* and *won* as well as their semantic link AGENT(*Russia*,*won*) as identified within the sentence *Russia won 397 medals at the Summer Olympic Games*:

```
<rdf:Description rdf:about="URL/7a1c9e/Token/0_5">
  <rdf:type rdf:resource="URL/terms/v1.0/Token"/>
  <ly:inSentence rdf:resource="URL/7a1c9e/S/0_53"/>
  <ly:text>Russia</ly:text>
  <ly:hasCountryName>russia</ly:hasCountryName>
  <ly:synsetId>108444912</ly:synsetId>
</rdf:Description>
<rdf:Description rdf:about="URL/7a1c9e/Token/7_9">
  <rdf:type rdf:resource="URL/terms/v1.0/Token"/>
  <ly:inSentence rdf:resource="URL/7a1c9e/S/0_53"/>
  <ly:text>won</ly:text>
  <ly:lemma>win</ly:lemma>
  <ly:isEvent>true</ly:isEvent>
  <ly:synsetId>201065094</ly:synsetId>
</rdf:Description>
<rdf:Description rdf:about="URL/7a1c9e/Token/0_5">
  <ly:isAgentOf rdf:resource="URL/7a1c9e/Token/7_9"/>
</rdf:Description>
```

where `ly` is defined in the XML namespace attribute `xmlns`, e.g., `<rdf:RDF xmlns:ly="URL/terms/v1.0/">`; `URL` may be `http://www.lymba.com/rdf` or the RDF generator's URL and `7a1c9e` is a sample hash value which uniquely identifies the document containing this sentence within the input collection, thereby ensuring the uniqueness of these URIs within the RDF store. In Figure 2, we show this sample representation in graphical form.

We note the representation of the named entity information as a predicate (`URI ly:hasCountryName "russia"` as apposed to `URI ly:hasNamedEntityType "country"`, which may require an additional triple pattern when queried using SPARQL, e.g. `URI ly:text "Russia"`).

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 (Figure 3). 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 creation of these entailment rules is a one-time process, which is independent of the RDF store chosen for this task. The resulting axioms can be loaded in the RDF store using the specific triple store's set of available commands for this purpose. The reasoning process is performed as soon as the triples capturing the documents' meaning are indexed in the RDF store.

The first variety of RDF rules involves the generation of *inSynset* triples between a document token and its corresponding ontological concept/WordNet synset. We note that by assigning word senses to each noun, verb, adjective, and adverb in our input documents, each of these tokens can be easily mapped to their corresponding WordNet synsets. The entailment rule that creates the *inSynset* triples is `(?tok ly:synsetId ?sid) (?syn wn:synsetId ?sid)` → `(?tok ly:inSynset ?syn)`.

Another type of RDF rules involves the generation of *owl:sameAs* predicates as coreferring entities within- as well as across-documents (e.g., all mentions of *Russia* refer to the same entity). Once these new triples are created, an OWLPrime [30] entailment will cluster all URIs of coreferring tokens, thus, gathering their corresponding semantic properties to a single URI. A sample rules used here is `(?tok1 ly:hasCountryName ?name) (?tok2 ly:hasCountryName ?name)` → `(?tok1 owl:sameAs ?tok2)`.

Semantic Calculus [3] rules are also converted into their corresponding RDF axioms to combine two semantic relation triples that share an argument. For

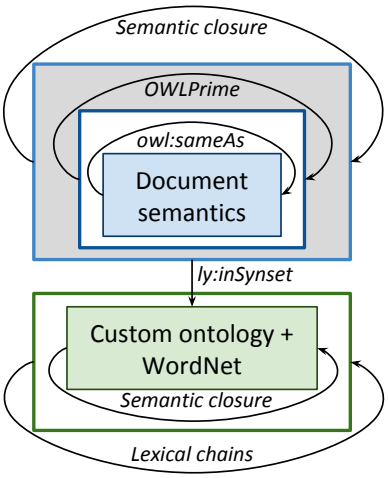

Figure 3: RDF store content after the reasoning step

instance, LOCATION(x, y) & PART-WHOLE(x, z) $\rightarrow$ LOCATION(z, y) can be defined as an RDF entailment rule `(?tok ly:isLocationOf ?lhs) (?rhs ly:isPartOf ?tok)` $\rightarrow$ `(?rhs ly:isLocationOf ?lhs)` to generate new LOCATION instances from existing LOCATION and PART-WHOLE triples. 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. These semantic combinations derive new information from a sentence such as *The country's finance minister, Ashraf Ghani, made the announcement at an international conference in Berlin, saying almost $4.5 billion has been promised for this year alone*, where a semantic parser may have already identified AGENT(*finance minister, made*), LOCATION(*international conference, made*), and LOCATION(*Berlin, international conference*). With these relations alone one cannot directly answer the question *Where is the finance minister?* (LOCATION(*finance minister, x*)). However, by combining these semantic relationships and generating new LOCATION triples (e.g., LOCATION(*international conference, finance minister*), LOCATION(*Berlin, finance minister*), and even LOCATION(*Germany, finance minister*)), the question can be answered with several degrees of specificity (*Berlin* or even *Germany*, by exploiting the PART-WHOLE relationship between these two concepts from WordNet). Other examples of Semantic Calculus axioms 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 [3].

Last, but not least, lexical chains[2] of WordNet relations can be translated to RDF entailment rules that generate new triples between semantically-linked concepts. For instance, *victim:n#1* $\leftarrow$ IS-A $\leftarrow$ *martyr:n#1* $\rightarrow$ DERIVATION $\rightarrow$ *martyr:v#1* $\rightarrow$ IS-A $\rightarrow$ *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?* Therefore, all lexical chains of maximum size 3 (up to 3 semantic links) are pre-computed and added to the RDF store as part of the ontological semantic model[3]. A sample lexical chain entailment rule is `(?a wn:hyponym ?b) (?b wn:derivation ?c) (?d wn:hyponym ?c)` $\rightarrow$ `(?a ly:isNearSynonymOf ?d).`

We note that this new body of knowledge is seamlessly blended into the RDF semantic index and is made available at query time. It greatly impacts the size and types of SPARQL triple patterns. It not only reduces the size of the SPARQL query and, thus, increasing the speed of returning an answer[4], but it also greatly improves the robustness of the SPARQL queries. More specifically, with no *isNearSynonymOf* triples, one would have to "guess" the semantic links between the question concepts in the answer documents. For instance, with no *isNearSynonymOf* for `?victim ly:inSynset ?syn . ?syn ly:isNearSynonymOf wn:synset-victim-noun-1`, the answer-yielding SPARQL query would have to include `?victim ly:inSynset ?syn . ?syn wn:hyponym ?s1 . ?s1 wn:derivation ?s2 . ?s2 wn:hypernym wn:synset-victim-noun-1`, which is a set of triple patterns very difficult to come by even for a human without having prior knowledge about the answer sentence. Furthermore, a single *isNearSynonymOf* triple accounts for up to three WordNet relation triples.

This RDF reasoning step (step 5), which employs all these types of axioms and completes the knowledge explicitly conveyed by the document content, finalizes the collection indexing phase of our SQA process.

[2] A WordNet lexical chain [8] is defined as a weighted path between two synsets, where the path consists of a sequence of two or more synsets connected by relations in WordNet.

[3] Each lexical chain has an associated weight, denoting the strength of the lexical chain, which is computed based on the type and direction of the chain's relations; e.g., (IS-A, IS-A) chain is stronger than a (IS-A, IS-A^{-1}) one, which is stronger than a (IS-A, PART-OF) chain.

[4] Despite having additional triples in the RDF store, large SPARQL queries take significantly longer to return an answer when compared to shorter ones. Also, if desired, certain triples can be removed from the RDF store once the reasoning process is completed in an effort to reduce the size of the triplestore.

2.2 Natural Language to SPARQL

Given our RDF-based QA framework, the question answering phase of the QA process must generate SPARQL 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. More specifically, the NL to SPARQL algorithm can be summarized as follows:

```
Input: Natural language question
Output: Precise answer

1 question understanding
  1.1 Deep NLP of input question
  1.2 answer type/answer type term
detection
2 SPARQL query formulation
3 query the RDF store
4 if answer(s) found, then 4.1, else 4.2
  4.1 return answers sorted by
confidence
  4.2 relax SPARQL query and go back
to 3
```

We note that, in order to be able to generate SPARQL queries that may identify answers within the RDF store, the NLP tools used to process the input question must aim to identify the same type of knowledge identified within input documents (in Figure 1, the Knowledge extraction and Question processing modules share the NLP). However, the 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 triple predicates, 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 identify the types of relations stored for the document content. For instance, if the RDF store contains ARRESTED_AT 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) & IS-A(y,*arrest*) $\rightarrow$ 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.

2.2.1 Question Processing

The first step (Step 1) 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 (Step 1.1), 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) (Step 1.2). We use 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 machine 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 [17, 18].

2.2.2 SPARQL Query Formulation

SPARQL queries are not easy to write even when one is 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. In Figure 4, we show a sample conversion of the question's semantics to SPARQL triple patterns that pinpoint the desired answer URI. We note that, for a complete SPARQL query, additional triple patterns must be added to identify the answer's lemma, sentence, or document information.

More specifically, in order to generate a SPARQL query, a unique SPARQL variable name is associated with each question token that 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 variables that denote the text of the sentence which contains that answer or the path of the answer's source document (e.g., `SELECT ?answer ?sentence ?path`).

The WHERE clause of the SPARQL query contains the semantic triples that must be satisfied by the indexed

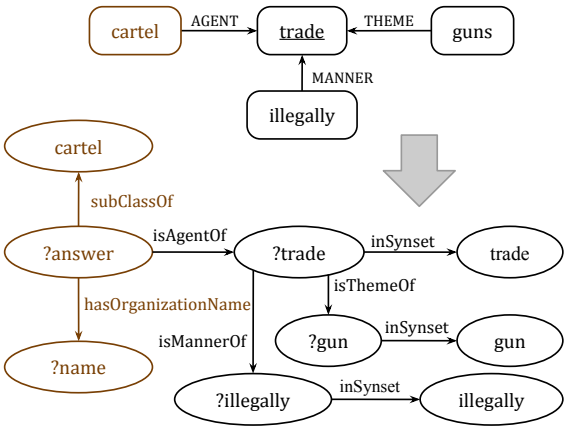

Figure 4: Semantics to SPARQL example

knowledge. First, the answer's semantic constraints are added:

1. Answer type filtering (e.g., for an *organization* answer type: `?ans ly:hasOrganizationName ?answer` – which forces the answer to be an organization named entity; if no answer type is identified, the query contains `?ans wn:hasLexeme ?answer`),

2. Answer type term tests (e.g., for the *cartel* answer type term: `?ans rdf:subCassOf wn:synset-cartel-noun-1 . ?ans ly:text ?answer` – which constrains the answer to hyponyms of the answer type term *cartel*), and

3. Sentence/document information (`?ans ly:inSentence ?sent . ?sent ly:text ?sentenceText` or `?ans ly:inDocument ?doc . ?doc ly:path ?path`).

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. `?ans ly:isAgentOf ?trade` for AGENT(*who*,*trade*); `?gun ly:isThemeOf ?trade` for THEME(*gun*,*trade*); `?illegally ly:isMannerOf ?trade` for MANNER(*illegally*, *trade*)). The other two describe the variables involved if not described already (e.g., `?trade ly:inSynset wn:synset-trade-verb-1`).

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

```
SELECT ?answer ?sentence ?path
WHERE {
  ?ans rdf:subCassOf wn:synset-cartel-noun-1 .
  ?ans ly:isAgentOf ?tr .
  ?tr ly:inSynset wn:synset-trade-verb-1 .
  ?gun ly:isThemeOf ?tr .
  ?gun ly:inSynset wn:synset-gun-noun-1 .
  ?il ly:isMannerOf ?tr .
  ?il ly:inSynset wn:synset-illegally-adverb-1 .
  ?ans ly:text ?answer .
  ?ans ly:inSentence ?sent .
  ?sent ly:text ?sentence .
  ?ans ly:inDocument ?doc .
  ?doc ly:path ?path
},
```

which can efficiently identify the correct answer from the RDF store.

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. However, these may not be identical to the concepts of the correct answer that are indexed in the RDF store. Therefore, if no rows are selected for this initial SPARQL query, we relax its constraints to allow for hyponyms, parts, derivations as well as near synonyms of the question terms to be matched by the RDF reasoning engine. For instance, `?trade ly:inSynset wn:synset-trade-verb-1` is replaced with `?trade inSynset ?syn . ?syn ly:isNearSynonymOf 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 constraints 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. Regardless, the relaxation criteria described in this section produces encouraging results as shown in Section 3.2.

2.2.4 Answer Ranking

By relaxing the original SPARQL query, if needed, the system is more likely to return more than one possible answer. Thus, an answer set ranking procedure is required before the results are presented to the user.

In order to understand why, an SQA system may return more than one answer to a single question, we

note that, for instance, the semantic relations that we index in the RDF data store may have originated from the content of the queried documents or may have been inferred from the original set of relations when they were combined using the Semantic Calculus entailment rules (Section 2.1.2). Thus, the triple patterns of a SPARQL query may find a match within each of these sets of relations. For instance, given the question *What is the area of operation for the Gulf Cartel?*, the SQA system extracts several LOCATION answers from the passage *It is also strong in Reynosa. Both cities are located in the Mexican state of Tamaulipas, which has become a bloody battleground between the Zetas and the Gulf Cartel*. These are: *Reynosa*, *Tamaulipas*, and *Mexico*, where LOCATION(*Reynosa, Gulf Cartel*) is identified in the content of the passage, and LOCATION(*Tamaulipas, Gulf Cartel*) and LOCATION(*Mexico, Gulf Cartel*) relations are derived from combinations with LOCATION(*Tamaulipas, Reynosa*)[5] and PART-WHOLE(*Tamaulipas, Mexico*)[6], both also extracted from the input text. Although the returned LOCATION relations are correct, they are not equally informative and, thus, should be ranked accordingly.

However, given the built-in architecture of triple-based RDF stores, the answers returned for a given SPARQL query are not sorted in any particular order. Even when multiple searches of the RDF store use the same query, the same answer set is returned in a completely different order. This impacts the overall quality of the responses returned by the SQA engine. To overcome this issue, we implemented an answer ranking module, which allows us to both sort and trim the returned results based on confidence scores.

The ordering of the answers to be returned is done at query time by the RDF store's internal retrieval engine by including an ORDER BY clause in our automatically generated SPARQL queries. The value used to sort the returned answers is the confidence of the semantic relation that links the answer concept to the rest of the question (e.g., LOCATION(x,*Golf Cartel*) for *What is the area of operation for the Gulf Cartel?*). However, in order for the RDF store to access the relation confidence information, this must be indexed in the data store when it is generated by the document indexing process. Therefore, the RDF representation of a semantic relation instance is changed to include the confidence assigned to it by the semantic parser. More specifically, the triple used to represent the semantic relation SR(x, y), `?x ly:sr ?y` is augmented with `?sr_uri ly:type ''sr''; ?sr_uri ly:lhs ?x; ?sr_uri ly:rhs ?y; ?sr_uri ly:confidence confidence.` For instance, given the sample RDF representation shown in Section 2.1.1, the RDF XML is generated to also include:

```
<rdf:Description rdf:about="URL/7a1c9e/SR/0_5-7_9">
  <rdf:type rdf:resource="URL/terms/v1.0/SR"/>
  <ly:type>isAgentOf</ly:type>
  <ly:confidence>0.856003</ly:confidence>
  <ly:lhs rdf:resource="URL/7a1c9e/Token/0_5"/>
  <ly:rhs rdf:resource="URL/7a1c9e/Token/7_9"/>
</rdf:Description>
```

For semantic relations extracted directly from text, the confidence value is the one assigned by the semantic parser. The confidence associated with a semantic relation generated by Semantic Calculus rules depends on the corresponding values of the relations that are being combined (the minimum confidence value of the input relations is assigned to the resulting relation). If $R_1(x, y) \& R_2(y, z) \rightarrow R_3(x, z)$,

$$conf(R_3) = \min(conf(R_1), conf(R_2)).$$

Optionally, each Semantic Calculus rule can be associated with its own confidence score, proportional with its accuracy of predicting the correct semantic relation on a large corpus [3]. The formula to compute the confidence of a resulting semantic relation will then become

$$conf(R_3) = conf(rule) * \min(conf(R_1), conf(R_2)).$$

We note that, with this triple scoring scheme, there is a preference towards relations identified in text or relationships derived from text relations within one or two inference steps because the reasoning process will propagate any errors of the semantic parser as well as any inaccuracies of the Semantic Calculus axioms.

The confidence values of the *isNearSynonymOf* triples is inversely proportional to the weight of the lexical chain that generated it. All *sameAs* and *inSynset* triples are not modified – are considered to have maximum confidence.

This change in the RDF representation of the semantic relations also prompts several modifications from the NL to SPARQL conversion module, which now makes use of the confidence values associated with the semantic relations of the answer. For the example shown in Section 2.2.2, the query changes to

```
SELECT ?answer ?sentence ?conf
WHERE {
  ?ans rdf:subCassOf wn:synset-cartel-noun-1 .
  ?sr ly:type "isAgentOf"^^xsd:string .
  ?sr ly:lhs ?ans .
  ?sr ly:rhs ?tr .
  ?sr ly:confidence ?conf .
```

[5] LOCATION(x, y) & LOCATION(y, z) $\rightarrow$ LOCATION(x, z); in this case: LOCATION(*Tamaulipas, Reynosa*) & LOCATION(*Reynosa, Gulf Cartel*) $\rightarrow$ LOCATION(*Tamaulipas, Gulf Cartel*).

[6] LOCATION(x, y) & PART-WHOLE(x, z) $\rightarrow$ LOCATION(z, y); here: LOCATION(*Tamaulipas, Gulf Cartel*) & PART-WHOLE(*Tamaulipas, Mexico*) $\rightarrow$ LOCATION(*Mexico, Gulf Cartel*).

```
  ?tr ly:inSynset wn:synset-trade-verb-1 .
  ?gun ly:isThemeOf ?tr .
  ?gun ly:inSynset wn:synset-gun-noun-1 .
  ?il ly:isMannerOf ?tr .
  ?il ly:inSynset wn:synset-illegally-adverb-1 .
  ?ans ly:text ?answer .
  ?ans ly:inSentence ?sent .
  ?sent ly:text ?sentence
},
... ORDER BY conf DESC
```

where the boldface triples have replaced the `?ans ly:isAgentOf ?tr` triple pattern to include the confidence value of the `isAgentOf` relation, which will be used to order the query's answers.

However, this new representation increases the size of the RDF store as well as the length of generated SPARQL queries (by 3 additional triple patterns; from an average of 5.33 triples to 8.16 for the experimental data described below, since only the confidence of the answer's semantic relation is used; all other semantic triple patterns remain unchanged). Nonetheless, there are two different types of triples within a SPARQL query: (1) Semantic triples that must be satisfied by an answer URI – they define the semantic graph to be matched within the RDF store (e.g., triples 1 through 10 in the SPARQL query shown above) and (2) Triples that identify the human-readable information about the answer URI that will be returned to a user (e.g., triples 11 and 13 in the same SPARQL query). In order to shorten the generated queries, we split the triple patterns into two different SPARQL queries: The first query retrieves the ranked list of answer URIs, and the second one requests each URI's lemma/sentence/document information, which is needed for display purposes. For instance,

```
SELECT ?ans ?conf
WHERE {
  ?ans rdf:subCassOf wn:synset-cartel-noun-1 .
  ?sr ly:type "isAgentOf"^^xsd:string .
  ?sr ly:lhs ?ans .
  ?sr ly:rhs ?tr .
  ?sr ly:confidence ?conf .
  ?tr ly:inSynset wn:synset-trade-verb-1 .
  ?gun ly:isThemeOf ?tr .
  ?gun ly:inSynset wn:synset-gun-noun-1 .
  ?il ly:isMannerOf ?tr .
  ?il ly:inSynset wn:synset-illegally-adverb-1
},
```

is the initial SPARQL query, which is followed by

```
SELECT ?answer ?sentence ?path ?conf
WHERE {
  ?ans ly:text ?answer .
  ?ans ly:inSentence ?sent .
  ?sent ly:text ?sentence .
  ?ans ly:inDocument ?doc .
  ?doc ly:path ?path
},
... ORDER BY conf DESC
```

where the `?ans` and `?conf` values are the ones returned by the first SPARQL query.

3 Experiments and Results

3.1 Data Sets and Triple Store Statistics

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 [1, 2]. 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 (when storing the semantic relation confidence values) or 3,840,980 triples for the simplified representation.

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 [1, 2]. Its RDF representation consists of 232,585 triples.

Other data sets that were used to ensure the robustness of our proposed RDF representation includes (1) the NIST TREC QA, a readily available open domain collection with known questions and answers (343 New York Times articles from June 1998 to September 2000; a total of 1.78MB of unstructured text; 172 articles contain the answers to 184 factoid questions used during the Question Answering (QA) Track of the 2006 Text REtrieval Conference (TREC) [17]; examples include: *How many people died in the tourist massacre at Luxor in 1997?, Who is the creator of The Daily Show?, What U.S. state is the highest avocado producer?, Where was the 1999 Sundance Film Festival held?*, etc.), and (2) the LDC Gigaword collection provided by IARPA's Knowledge Discovery and Dissemination (KDD) program [10] (2,979 documents pertaining to the programs's tasks; 8.56MB of unstructured text; example questions: *Find associates of Abderraouf Jdey., Find locations of suicide bombings in 2005., Find people born in Mosul, Iraq.*).

For all these data sets, we tracked the various number of triples generated by the reasoning methods proposed in this paper (Section 2.1.2). In Table 1, we show these values for all document collections. We note that the Semantic Calculus inferences did not complete on the Illicit Drugs data set. Individual rule evaluations were performed to determine inaccurate rules. However, the process of generating new semantic relations using one entailment rule at a time does not provide a clear understanding of how the rules interact when used together, especially on a large data set. The task became tractable on the TREC QA

Table 1: RDF store content/size for various data sets

RDF Store	Illicit Drugs	TREC QA	LDC Gigaword	LDC Gigaword w/ relation confidence
Basic model	3,840,980	1,934,115	8,807,032	16,035,706
sameAs links	761,139 (cross-document)	217,362 (cross-document)	107,162 (within-document only)	
Coreference model	3,693,153	1,703,262	7,924,644	15,432,526
Semantic Calculus	-	222,978	(not-required)	
inSynset triples	229,427	94,375	387,149	
WordNet	1,814,641			
Illicit Drugs ontology	232,585	(not applicable)	(not applicable)	
Lexical Chains	1,604,112 (length < 3); 40,837,911 (length < 4)			
Final model	7,573,918	5,439,368	11,730,546	19,238,428

RDF store, where only 222,978 semantic relations with a confidence greater than 0.6 were generated and, therefore, retained in the RDF store. For the Phase 2 of the KDD program, the semantics of interest were limited to several very specific domain relations, which were defined by combining the core semantic relations (e.g., AGENT, THEME, LOCATION, IS-A, PART-WHOLE, KINSHIP, etc.) using Semantic Calculus rules prior to the RDF representation of the input, which was modified to store only the set of custom relationships (e.g., COLLEAGUE, RELATED_EVENT, HAS_SKILL, LEADS_GROUP, etc.), which are the focus of the questions for this data set. Another characteristic of this data set is its high frequency of name mentions across documents, which caused the cross-document co-reference resolution procedure to fail to terminate. For the LDC Gigaword data set, there are 27,594,179 *sameAs* triples between all its concepts. Additionally, we show in Table 1 the impact on the size of the RDF store that the changes in the RDF representation of semantic relations have when retaining the relation's confidence value information.

In summary, the contents of the RDF store can be highly customized depending on the task at hand as well as the nature of the data collection. The base model can be trimmed to include only the semantics of interest. The OWLPrime entailment step can be omitted if too many *sameAs* triples are generated. But, despite having to disable certain modules for various data collections, the type of inferences listed in Section 2.1.2 complement each other and become vital to the semantic question answering process.

We also note that the SQA system was also applied to monitoring treatment efficiency reported in biomedical papers from PubMed [21]. It stores the full semantic representation of 7 million documents and provides real-time QA for this collection of biomedical articles. No formal quality evaluation is currently available for this data set.

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 [29, 6], 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.

If none of the returned answers are correct, we use a mean reciprocal rank of 0. For questions with multiple correct answers in the result set (e.g., list questions), we use the rank of the first correct answer.

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 [17, 18].

The set of questions used to evaluate the SQA system varies 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 tweezes?*) – 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 Methamphetamines?), 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. Our query relaxation procedures preserve certain semantic constraints within a SPARQL query. However, at the extreme, a SPARQL query could be relaxed to return all sentences that include the question's key terms and an entity that matches the question's answer type, with no other semantic restrictions. Within this setting, the SQA system becomes similar to a free-text search index-based system, which bases its passage retrieval on keywords and named entity types. Alternatively, the SQA engine can back-off to return the results of a free-text search index, when no answer can be retrieved using semantics.

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 to *isNearSynonymOf*) was sufficient for 30.98% of the test SPARQL queries. For remaining queries, one or more semantic relation triple patterns were dropped from the SPARQL query in an effort to identify an answer within the RDF store.

3.2.3 Error Analysis

Given the complexity of our SQA engine, we analyzed the errors it made and their sources in order to determine the systems shortcomings and possible future improvements. For this purpose, an answer set is deemed inaccurate if the correct answer is not among the top five results.

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. We include in this percentage value the errors made when the SQA system failed to find any answers in the RDF data store.

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 performance of the system drops to 53.16% MRR when we disable the answer ranking module. All errors caused by incorrect SPARQL queries due to inaccurate/missing semantic relations were included in the previous category of system mistakes.

Last but not least, the addition of three Semantic Calculus entailment rules would have closed the semantic gap between concepts that, otherwise, matched the corresponding SPARQL queries of five test

questions. Without these rules, the system was not able to identify the correct answers.

4 CONCLUSIONS

In this paper, we describe a language-independent RDF triple-based semantic question answering framework. Within this framework, the rich semantic structures identified in unstructured data sources are stored into scalable RDF stores. Furthermore, this framework 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. We present an optimal RDF representation for the extracted semantic information as well as various natural language reasoning methodologies for the generated RDF store. We detail, not only the means to automatically create robust SPARQL queries, but also query relaxation procedures and an answer ranking technique, which are vital when dealing with the imperfect semantics identified by a machine. We show how the SQA framework can be employed for various tasks and domains without a loss in capabilities. This semantic approach to question answering yields more accurate results when compared with a free-text search index-based question answering engine.

ACKNOWLEDGEMENTS

We would like to thank GRAPHIQ (www.graphiq.com/), which sponsors the open-access publishing of this paper.

REFERENCES

[1] M. Balakrishna and D. Moldovan, "Automatic Building of Semantically Rich Domain Models from Unstructured Data," in *Proceedings of the Twenty-Sixth International Florida Artificial Intelligence Research Society Conference (FLAIRS 2013)*, St. Pete Beach, Florida, May 22-24, 2013.

[2] M. Balakrishna, D. Moldovan, M. Tatu, and M. Olteanu, "Semi-Automatic Domain Ontology Creation from Text Resources," in *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC 2010)*, Valletta, Malta, May 17-23, 2010.

[3] E. Blanco and D. I. Moldovan, "Unsupervised Learning of Semantic Relation Composition," in *Proceedings of Human Language Technology*, 2011, pp. 1456–1465.

[4] A. Bouziane, D. Bouchiha, N. Doumi, and M. Malki, "Question Answering Systems: Survey and Trends," *Procedia Computer Science*, vol. 73, pp. 366 – 375, 2015.

[5] D. Damljanovic, M. Agatonovic, and H. Cunningham, "FREyA: An Interactive Way of Querying Linked Data Using Natural Language," in *Proceedings of 8th Extended Semantic Web Conference*, 2012, pp. 125–138.

[6] H. T. Dang, D. Kelly, and J. Lin, "Overview of the TREC 2007 Question Answering Track," in *Proceedings of The Sixteenth Text REtrieval Conference*, 2008.

[7] "DBpedia," http://wiki.dbpedia.org/, accessed: June 16, 2016.

[8] T. Erekhinskaya and D. Moldovan, "Lexical Chains on WordNet and Extensions," in *Proceedings of the Twenty-Sixth International Florida Artificial Intelligence Research Society Conference (FLAIRS 2013)*, C. Boonthum-Denecke and G. M. Youngblood, Eds. AAAI Press, 2013.

[9] C. Fellbaum, Ed., *WordNet: An Electronic Lexical Database*. Cambridge, MA: MIT Press, 1998.

[10] IARPA, "Knowledge Discovery and Dissemination (KDD)," https://www.iarpa.gov/index.php/research-programs/kdd, accessed: June 16, 2016.

[11] E. Kaufmann, "Talking to the Semantic Web? Natural Language Query Interfaces for Casual End-users," Ph.D. dissertation, University of Zurich, February 2009. [Online]. Available: http://www.ifi.uzh.ch/pax/web/uploads/pdf/publication/1202/Dissertation_Esther_Kaufmann.pdf

[12] T. Khot, N. Balasubramanian, E. Gribkoff, A. Sabharwal, P. Clark, and O. Etzioni, "Markov Logic Networks for Natural Language Question Answering," *Computing Research Repository (CoRR)*, vol. abs/1507.03045, 2015.

[13] K. Liu, J. Zhao, S. He, and Y. Zhang, "Question Answering over Knowledge Bases," *IEEE Intelligent Systems*, vol. 30, no. 5, pp. 26–35, 2015.

[14] V. Lopez, M. Fernndez, E. Motta, and N. Stieler, "PowerAqua: Supporting users in querying and exploring the Semantic Web." *Semantic Web*, vol. 3, no. 3, pp. 249–265, 2012.

[15] 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*, vol. 5, no. 2, 2007.

[16] V. Lopez, V. Uren, M. Sabou, and E. Motta, "Is Question Answering Fit for the Semantic Web?: A Survey," *Semantic Web*, vol. 2, no. 2, pp. 125–155, Apr. 2011.

[17] D. Moldovan, M. Bowden, and M. Tatu, "A Temporally-Enhanced PowerAnswer in TREC 2006," in *Proceedings of Text REtrieval Conference*, 2006.

[18] D. Moldovan, C. Clark, and M. Bowden, "Lymba's PowerAnswer 4 in TREC 2007," in *Proceedings of Text REtrieval Conference*, 2007.

[19] Oracle, "RDF Semantic Graph Prerequisites, and Advanced Performance and Scalability for Semantic Web Applications," 2014.

[20] A.-M. Popescu, O. Etzioni, and H. Kautz, "Towards a Theory of Natural Language Interfaces to Databases," in *Proceedings of 2003 International Conference on Intelligent User Interfaces (IUI'03)*, 2003, pp. 149–157.

[21] "PubMed," http://www.ncbi.nlm.nih.gov/pubmed, accessed: June 16, 2016.

[22] RDF Working Group, "Resource Description Framework (RDF)," http://www.w3.org/RDF/, 2014.

[23] M. Richardson and P. Domingos, "Markov Logic Networks," *Mach. Learn.*, vol. 62, no. 1-2, pp. 107–136, Feb. 2006.

[24] M. Tatu, M. Balakrishna, S. Werner, T. Erekhinskaya, and D. Moldovan, "Automatic Extraction of Actionable Knowledge," in *Proceedings of IEEE Tenth International Conference on Semantic Computing*, 2016.

[25] M. Tatu, S. Werner, M. Balakrishna, T. Erekhinskaya, and D. Moldovan, "Semantic Question Answering on Big Data," in *Proceedings of International Workshop on Semantic Big Data (SBD 2016)*, 2016.

[26] The Apache Software Foundation, "Apache Jena - A free and open source Java framework for building Semantic Web and Linked Data applications," http://jena.apache.org/, accessed: June 16, 2016.

[27] 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*, 2012, pp. 639–648.

[28] 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 *Cross-Language Evaluation Forum CLEF (Working Notes)*, ser. CEUR Workshop Proceedings, L. Cappellato, N. Ferro, G. J. F. Jones, and E. SanJuan, Eds., vol. 1391, 2015.

[29] E. M. Voorhees, "The TREC-8 Question Answering Track Report," in *Proceedings of the Eighth Text REtrieval Conference*, 1999, pp. 77–82.

[30] W3C, "Oracle OWLPrime," www.w3.org/2007/OWL/wiki/OracleOwlPrime, Jan 24, 2008.

[31] W3C, "SPARQL Query Language for RDF," http://www.w3.org/TR/rdf-sparql-query/, January 15, 2008.

[32] "WordNet RDF," http://wordnet-rdf.princeton.edu/, November 7, 2013.

[33] M. Yahya, K. Berberich, S. Elbassuoni, M. Ramanath, V. Tresp, and G. Weikum, "Natural Language Questions for the Web of Data," in *Proceedings of the 2012 Conference on Empirical Methods on Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL '12)*, 2012, pp. 379–390.

[34] X. Yao, J. Berant, and B. Van Durme, "Freebase QA: Information Extraction or Semantic Parsing?" in *Proceedings of ACL Workshop on Semantic Parsing*, 2014.

APPENDIX: ADDITIONAL NL TO SPARQL EXAMPLE

The conversion from NL to SPARQL for the input natural language query *Find people who have met with Abu Khabab in Afghanistan.* is graphically shown in Figure 5. This LDC Gigaword query makes use of custom semantic relations (e.g., MEETS-WITH), which are generated before attempting to create the SPARQL query. We note that the core AGENT, and THEME relationships are used to generate new instances of semantic relations (dashed blue lines in Figure 5). However, they do not produce semantic triple patterns for the SPARQL query. We note that, for Arabic names, a normalized value is used, more specifically, the name's phonetic representation (e.g., *Al Qaeda*, *al-Qaida*, and *al-Qaida* are all normalized to *al kaida*).

The resulting SPARQL query is:

```
SELECT ?answer ?sentenceText
WHERE {
  { ?ans rdf:subCassOf wn:synset-people-noun-1 }
  UNION { ?ans ly:hasHumanName ?answer } .
  ?ans ly:meetsWith ?abu .
  ?abu ly:hasHumanName "abu kabab"^^xsd:string .
  ?ans ly:atLocation ?afghan .
  ?afghan ly:inSynset wn:synset-afghanistan-noun-1 .
  ?ans ly:sentence ?sentenceText
}.
```

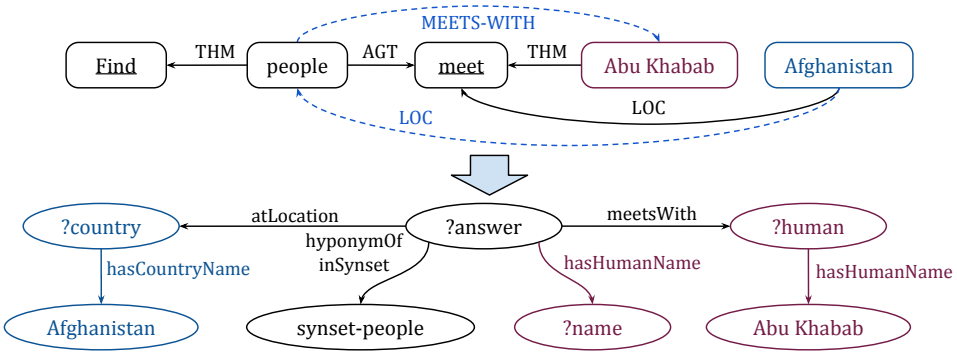

Figure 5: Semantics to SPARQL example for the KDD dataset

AUTHOR BIOGRAPHIES

Marta Tatu received her PhD in Computer Science from The University of Texas at Dallas in 2007. As a research scientist at Lymba Corporation, she focuses on applying knowledge extraction and question answering technologies to various unstructured data sources such as contracts, scientific proposals, and research publications. She has been instrumental in the development of Lymba's state-of-the-art question answering system, PowerAnswer, which topped NIST TREC QA evaluations for seven years. As the PI of an NSF award, she has focused on developing a solution to process large amounts of unstructured data. By aligning the extracted knowledge against a common ontology, the solution provides customized semantic search to intelligence analysts.

Mithun Balakrishna received his PhD in Computer Science from The University of Texas at Dallas in 2007. His academic research focused on extraction and application of high-level linguistic knowledge to improve spoken language recognition and understanding. He currently leads Lymba Corporation's research and business thrusts in the area of Knowledge Engineering and Management. His research is focused on the development of tools to automatically build semantically rich knowledge models for specific domains using relevant unstructured data. He has been the PI on several enterprise and government projects, including Spoken Dialog Question Answering, Automatic Extraction of Biographical Profiles, Ontologies for Education, and Antibiotic Resistance Knowledge Extraction.

Steven Werner received his bachelor's degree in Computer Science in 2013 from The University of Texas at El Paso. His research as a member of the Interactive Systems Group at UTEP focused on unsupervised techniques for spoken information retrieval. He joined Lymba Corporation in May 2014. He has focused his efforts on classification projects for the Customer Relationship Management vertical, as well as knowledge extraction projects for multiple domains, including intelligence, medical, financial, and retail. He has expertise in semantic search and natural-language querying. His current research interests include deep semantic processing, Linked Data and ontologies.

Tatiana Erekhinskaya studied Mathematics at the Nizhniy Novgorod State University, Russia. She worked at Dictum Ltd focusing on natural language processing of the Russian language. The key projects include syntactic parser robust to spelling mistakes and opinion mining using dependency parsing. She received her MBA degree in 2010. She received her PhD in Computer Science from The University of Texas at Dallas in 2014. Her dissertation focused on probabilistic models for text understanding. Today, she is a research scientist at Lymba Corporation. Her primary areas of research are: deep semantic processing with special focus in medical domain and big data.

Dan Moldovan received his diploma degree in Engineering in 1969 from the Polytechnic Institute of Bucharest, Romania. He received his PhD in Computer Science in 1978 from Columbia University, New York. He is a Professor of Computer Science at the University of Texas at Dallas and the co-Director of the Human Language Technology Research Institute at UTD. Previously he held faculty positions at the University of Southern California and Southern Methodist University in Dallas. He was a Program Director at NSF while in sabbatical from USC. He is the Founder and Chief Scientist of Lymba Corporation, a Texas based company specializing in Natural Language Processing products and solutions. His current research interests are in lexical semantics, in particular studying the representation and reasoning of explicit and implicit text knowledge, and transforming unstructured data into semantic triples.