QASYO: A Question Answering System for YAGO Ontology

Abdullah M. Moussa and Rehab F. Abdel-Kader
Electrical Engineering Department, Faculty of Engineering - Port-Said,
Port-Said University, Port Fouad 42523, Port-Said, Egypt
E-mail: abdallah.elsayed87@gmail.com, r.abdelkader@eng.psu.edu.eg

Abstract

The tremendous development in information technology led to an explosion of data and motivated the need for powerful yet efficient strategies for data mining and knowledge discovery. Question Answering (QA) systems made it possible to ask questions and retrieve answers using natural language (NL) queries, rather than the keyword-based retrieval mechanisms used by current search engines. In Ontology-based QA system, the knowledge based data, where the answers are sought, has a structured organization. The question-answer retrieval of ontology knowledge base provides a convenient way to obtain knowledge for use, but the natural language need to be mapped to the query statement of ontology. QASYO is a sentence level question-answering system that integrates natural language processing, ontologies and information retrieval technologies in a unified framework. It accepts queries expressed in natural language and YAGO ontology as inputs and provides answers drawn from the available semantic markup. QASYO combines several powerful techniques in a novel way to make sense of NL queries and to map them to semantic markup. Semantic analysis of questions is performed in order to extract keywords used in the retrieval queries and to detect the expected answer type. In this paper we describe the current version of the system, in particular discussing its reasoning capabilities, and Performance.

Keywords: *Information Retrieval; Question Answering; Ontologies; YAGO.*

1. Introduction

The increase in web popularity has created the demand for systems that help the users find relevant information easily. One example is question answering systems that appear with the aim of providing precise textual answers to specific natural language users' queries rather than a set of matching Web documents [1]. The purpose of a QA system is to find the correct answers to user arbitrary questions in both non-structured and structured collection of data. Automated question answering (QA) research can be dated back to the 1960s; however it has often been confined to domain-specific expert systems [1-2]. Researchers have experimented with QA systems based on closed, pre-tagged corpora [3] or knowledge bases [4 -5]. On the web, a typical example of a QA system is Jeeves [6] which allows the users to ask questions in natural language. It looks up the user question in its own database and returns a list of matching question which it knows to answer. The user selects the most appropriate question from the list.

Most existing information technology applications use only a single source of background knowledge mostly WordNet or Wikipedia. They could significantly improve their performance, if a huge ontology with knowledge from several sources was available [7- 12]. Ontology formally defines a conceptual representation of concepts and their relationships within a specific domain. Ontologies play an essential role in the semantic web by enabling knowledge sharing and exchange. However, as stated in Bernstein et al. [13] enabling non-expert users to exploit this information means bridging the gap between the logic-based semantic representation of the information and the natural language expressiveness employed by users. Ontology is a key factor in building the Semantic Web, where it is easier for the machines to communicate the information. However, today research in ontology is far beyond the Semantic Web; ontology can be seen as a knowledge base which has a more complex form than a relational database. Many applications in modern information technology utilize ontological background knowledge [8, 9, 14-16]. This applies above all to applications in the vision of the semantic web, but there are many other application fields. Machine translation, and document classification based on supervised or semi-supervised learning can be combined with ontologies. Furthermore, ontological knowledge structures play an important role in the utility of background knowledge for question answering and information retrieval.

In particular, a knowledge based QA system can help with answering questions requiring situation specific knowledge, where multiple pieces of information need to be inferred and combined at run time, rather than simply having a pre-written paragraph of text retrieved [11-12]. In the case of an ontology-based QA system, knowledge based data, where the answers are sought, has a structured organization defined by an ontology. Users could raise questions in natural language and the system will return accurate answers to users directly after question analyzing, information retrieval and answer extraction. The semantic web vision is one in which rich, ontology-based semantic markup is widely available, thus opening the way to novel, sophisticated forms of question answering. Ontology knowledge base provides a convenient way to obtain knowledge for users, but the natural language need to be mapped to the query statement of ontology. Prior investigations have shown that there are two major challenges in the ontological QA systems that need to be addressed. First, understanding the intention of user question in question analyzing phase and representing it in a formal way. Second, translating this formal representation into a correct query adapted to the formal underlying ontology query schema. The first issue can be resolved through natural language processing technology, while the second is crucial for ontology retrieval to get answer.

Recently, researchers have been attracted to the task of developing Ontology based QA systems such as Querix [17], and AquaLog [11, 12]. Querix introduced by Kaufmann et al. is another ontology-based question answering system that translates generic natural language queries into SPARQL. In case of ambiguities, Querix relies on clarification dialogues with users. In this process users need to disambiguate the sense from the system-provided suggestions. AquaLog Proposed by Lopez et al. [11, 12] is a portable semi-automatic QA system that combines natural language processing, ontologies, logic, and information retrieval technologies. We say that AquaLog is portable, because the configuration time required to customize the system for a particular ontology is negligible. In AquaLog the ontology is used intensively as it is utilized in the refinement of the initial query, the reasoning process, and in the similarity algorithm AquaLog provides the best balance between domain customization effort and performance. It is also backed by a learning mechanism, so that its performance improves over time, in response to the vocabulary used by the users. AquaLog adopts an intermediate representation of user-defined triad, classifies the problem manually, but user involvement is necessary in identification process of multi-candidate concepts and relationships. However, this system heavily relies on language processing and requires syntactically correct sentences.

In our work we use the YAGO [7, 8, 18] ontology as the background knowledge which integrates the conceptual hierarchy of the WordNet dictionary with the high coverage of Wikipedia. YAGO is the first known approach that accomplishes this unification between WordNet and facts derived from Wikipedia with an accuracy of 95%. This allows the YAGO ontology to profit, on one hand, from the vast amount of individuals known to Wikipedia, while exploiting, on the other hand, the clean taxonomy of concepts from WordNet. Currently, YAGO contains roughly 1.7 million entities and 15 million facts about them. We have already mentioned that many systems simply use ontology as a mechanism to support query expansion in information retrieval. In contrast with these systems QASYO is interested in providing answers derived from semantic annotations to queries expressed in NL.

QASYO is a question-answering system which takes queries expressed in natural language and ontology as inputs and returns answers drawn from the available ontology-compliant semantic markup. QASYO combines several powerful techniques in a novel way to make sense of NL queries and to map them to semantic markup. Specifically, it makes use of WordNet [19, 20], and novel ontology-based services for relations and classes to make sense of user queries with respect to the target knowledge base. In this paper we describe the current version of the QASYO system, in particular discussing its reasoning capabilities, and performance. The rest of this paper is organized as follows: YAGO ontology is described in Section 2. In Section 3, we present the QASYO architecture and it's Linguistic Component. The experimental evaluation is reported in Section 4. Finally, we summarize the paper with some concluding remarks and future research directions in Section 5.

2. YAGO Ontology

YAGO is a new light-weight ontology that combines high coverage with high quality [7, 8]. It integrates the conceptual hierarchy of the WordNet dictionary with the high coverage of Wikipedia one of the most comprehensive lexicons available today. WordNet is one of the mostly widely used background knowledge in modern information technology. WordNet is a semantic lexicon for the English language developed at the Cognitive Science Laboratory of Princeton University. It distinguishes between words as literally appearing in texts and the actual senses of the words. A set of words that share one sense is called a synset. Thus, each synset identifies one sense (i.e., semantic concept). Words with multiple meanings (ambiguous words) belong to multiple synsets. WordNet's latest version is 3.0. As of 2006, the database contains 155,287 words organized in 117,659 synsets for a total of 206,941 word-sense pairs; in compressed form, it is about 12 megabytes in size. WordNet distinguishes between nouns, verbs, adjectives and adverbs because they follow different grammatical rules. WordNet provides relations between synsets such as hypernymy/hyponymy (i.e., the relation between a sub-concept and a super-concept) and holonymy/meronymy (i.e., the relation between a part and the whole).

Wikipedia is a well-known multilingual, web-based, free-content encyclopedia. It is written collaboratively by volunteers and is considered one of the most comprehensive lexicons available today. The English version of Wikipedia in January 2007 comprised 1,600,000 articles at that time. Each Wikipedia article is a single Web page and usually describes a single topic. The majority of Wikipedia pages have been manually assigned to one or multiple categories. The page about Albert Einstein, for example, is in the categories German language philosophers, Swiss physicists, and 34 more. Conveniently, the categorization of Wikipedia pages and their link structure are available as SQL tables, so that they can be exploited without parsing the actual Wikipedia articles. But rather than using information extraction methods to leverage the knowledge of Wikipedia, YAGO approach utilizes the fact that Wikipedia has category pages. Category pages are lists of articles that belong to a specific category (e.g., Zidane is in the category of French football players). These lists give us candidates for entities (e.g. Zidane), candidates for concepts (e.g. IsA(Zidane, FootballPlayer)) [15] and candidates for relations (e.g. ISCITIZENOF(Zidane, France)). In ontology, concepts have to be arranged in taxonomy to be of use. The Wikipedia categories are indeed arranged in a hierarchy, but this hierarchy is barely useful for ontological purposes. For example, Zidane is in the super-category named "Football in France", but Zidane is a football player and not a football. In contrast, WordNet provides a clean and carefully assembled hierarchy of thousands of concepts. But the Wikipedia concepts have no obvious counterparts in WordNet. YAGO can accomplish the unification between WordNet and facts derived from Wikipedia with an accuracy of 95%. This allows the YAGO ontology to profit, on one hand, from the vast amount of individuals known to Wikipedia, while exploiting, on the other hand, the clean taxonomy of concepts from WordNet.

As in RDF [21] and OWL [22], all objects (e.g. cities, people, even URLs) are represented as *entities* in the YAGO model. Two entities can stand in a *relation*. YAGO has dozens of relations, and the relations in YAGO are consist of two types: taxonomic relations such as "SUBCLASSOF", "TYPE", and "MEANS" and non-taxonomic relations such as "HASWONPRIZE", "FAMILYNAMEOF", "GIVENNAMEOF" and so on. Table 1 presents some of the largest relations in YAGO. The triple <entity, relation, entity> is called a *fact*. YAGO consists of more than 1.7 million *entities* (like books, countries, movies, persons, geopolitical etc.) and over 14 million *facts* about them. Tables 2 and 3 present the sizes of YAGO entities and the sizes of other ontologies respectively.

Table 1. Example of Largest Relations in YAGO

Relation	Domain	Range	# Facts
SUBCLASSOF	class	class	211,979
TYPE	entity	class	3,957223
BORNONDATE	person	date	350,613
DIEDONDATE	person	date	168,037
ESTABLISHEDONDATE	entity	date	69,529
LOCATEDIN	object	region	125,738
WRITTENINYEAR	book	year	16,441
HASWONPRIZE	person	prize	13,645
MEANS	word	entity	4,014,819
FAMILYNAMEOF	word	person	466,969
GIVENNAMEOF	word	person	464,816

Table 2. Size of YAGO Entities

Relations	92
Classes	224, 391
Individuals	1,531,588

Table 3. Size of Other Ontologies

Ontology	# Entities	# Facts
WordNet 3.0 [19, 20]	117,659	821,492
Cyc[23]	300,000	3,000,000
YAGO[7, 8, 18]	1,700,000	15,000,000
DBpedia[24]	1,950,000	103,000,000

The facts have been automatically extracted from Wikipedia and unified with WordNet, using a carefully designed combination of rule-based and heuristic methods. The resulting knowledge base is a major step beyond WordNet: in quality by adding knowledge about individuals like persons, organizations, products, etc. with their semantic relationships – and in quantity by increasing the number of facts by more than an order of magnitude. The empirical evaluation of fact correctness shows an accuracy of about 95%. YAGO is based on a logically clean model, which is decidable, extensible, and compatible with RDFS. For each general query, we can find the corresponding category concepts through taxonomic relations and for each concept we should find all the fact information of the concept through the relations.

Let's take "Elvis" for example, for the query "Elvis", we can find one *word entities* "Elvis" in YAGO, so we can get facts about the entities through non-taxonomic relation For example, to state that Elvis Presley won the Grammy Award, we say that the entity Elvis Presley stands in the HASWONPRIZE relation with the entity Grammy Award. We write

Elvis Presley HASWONPRIZE Grammy Award

Similarly to state that Elvis has family name of Presley we write:

"Elvis" FAMILYNAMEOF Presley

Numbers, dates, strings and other literals are represented as entities as well. This means that they can stand in relations to other entities. For example, to state that Elvis Presley was born in 1935 we write:

Elvis Presley BORNINYEAR 1935

Entities are abstract ontological objects, which are language-independent in the ideal case. Language uses words to refer to these entities. In the YAGO model, words are entities as well. This makes it possible to express that a certain word refers to a certain entity, like in the following example: Elvis is a person entity, and we can refer it as a people concept. Also we can find the taxonomic relation of the entities: MEANS, TYPE and so on.

"Elvis" MEANS Elvis Presley

This allows us to deal with synonymy and ambiguity. The following line says that "Elvis" may also refer to the English songwriter Elvis Costello:

"Elvis" MEANS Elvis Costello

We use quotes to distinguish words from other entities. Similar entities are grouped into *classes*. For example, the class physicist comprises all physicists and the class word comprises all words. Each entity is an *instance* of at least one class. We express this by the type relation:

Elvis Presley TYPE singer

Classes are entities that are arranged in a taxonomic hierarchy, expressed by the SUBCLASSOF relation:

singer SUBCLASSOF person

Ontology triplet is a triplet consisting of two classes and their relations, such as (has-CEO, Company, Person). The triple of an entity, a relation and an entity is called a fact. The two entities are called the *arguments* of the fact. Each fact is given a *fact identifier*. As RDFS, the YAGO model considers fact identifiers to be entities as well. This allows us to store meta-relations uniformly together with usual relations. For instance, for each individual we can store the URL of the corresponding Wikipedia page. This will allow future applications to provide the user with detailed information on the entities. For example, suppose that the above fact (Albert Einstein, BORNINYEAR, 1879) had the fact identifier #1, and then the following line would say that this fact was found in Wikipedia:

#1 FOUNDIN http: //www.wikipedia.org/Einstein

A query engine for accessing the content of YAGO is provided by the YAGO creators. The YAGO query engine processes only YAGO compatible query-triples such as (Einstein HASDOCTORALADVISOR $x). Table 4 shows two simple queries on the YAGO ontology. The second query makes use of the distinction between words and other individuals in YAGO.

Table 4. Simple Queries on YAGO

Query	Result
Who was Einstein's doctoral advisor? Einstein HASDOCTORALADVISOR $x	$x=Alfred Kleiner
Who is named after a place in Africa? $place LOCATEDIN Africa $name MEANS $place $name FAMILYNAMEOF $who	$who=Gabriel Sudan and 22 more

The YAGO model itself is independent of a particular data storage format. To produce minimal overhead, simple text files are used as an internal format. A folder is maintained for each relation and each folder contains files that list the entity pairs. Furthermore, conversion programs are available to convert the ontology to different output formats. First, YAGO is available as a simple XML version of the text files. Furthermore, YAGO can be converted to a database table. The table has the simple schema FACTS (factId, arg1, relation, arg2, confidence). Several software interfaces are available to load YAGO into Oracle, or MySQL databases.

3. QASYO System Architecture

This section presents the architecture and functionality of QASYO. In the QASYO model there are 4 phases: question classifier, linguistic component, query generator and query processor. The QASYO architecture can be characterized as a waterfall model, during which a NL query gets translated into a set of intermediate, triple-based representations, query-triples, and then these are translated into ontology-compatible triples, as shown in Figure 1. There are two main reasons for adopting a triple-based data model: first of all, it is possible to represent most queries as triples. Secondly, RDF-based knowledge representation formalisms for the semantic web, such as RDF itself [21] or OWL [22] also subscribe to this binary relational model and express statements as <subject, predicate, object>. Hence, it makes sense for a query system targeted at the semantic web to adopt this data model.

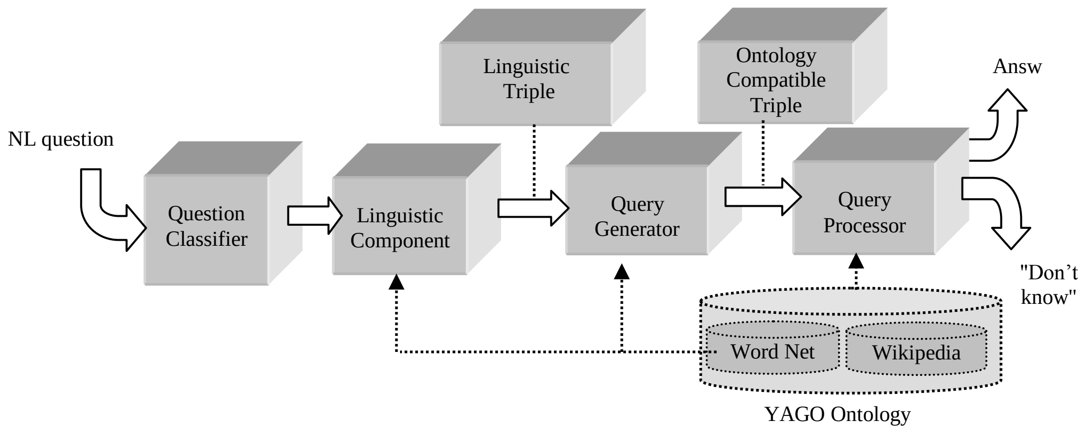

Figure 1. The QASYO Architecture.

The QASYO system takes as input the domain ontology populated with domain data, and an informal query expressed in natural language. As output, the information required in the query is returned to the user. The whole QA process is composed of two consecutive phases: question analysis and answer retrieval.

- Question analysis. The question analysis phase processes a natural language query in order to obtain a formal representation to be compliant with the YAGO-query formulation database. This requires several stages such as classifying and parsing the question. The output of the question analysis is a query pattern. That is to say, a natural language query labelled with morphological information and ontology concepts.

- Answer retrieval. the input query pattern is processed by the seach engine used by the ontological database. This phase returns the required answer for the user question or a simple "Don't know" message.

3.1. Question Classifier

The question classification phase classifies the question as one of the types supported by QASYO including: basic queries requiring an affirmation/negation and W/H questions (what, who, where, when, and which). Most QA systems use question type to judge the answer. They classify the question types on the basis of types of answers users are expecting. For example, a "Who is …?" will be assigned a "PERSON/ORGANIZATION" type; while "When did … happen?" will be classified as "DATE/TIME" question.
The *expected answer type* is the semantic type of the entity expected as the answer to the question. The expected answer type is helpful for factual questions. The answer type can be a combination of categories; like in the case of "*Who killed John Fitzgerald Kennedy*?" the answer type is person or organization. Categories can often be determined directly from a W/H-word: "*who*", "*when*", "*where*". The category for "*how <adjective>*" is determined from the adjective category: "*many*", "*much*" for quantity, "*long*", "*old*" for time, etc. The type for a "*what <noun>*" question is normally the semantic type of the noun. For instance in "*what king signed the MagnaCharta*?" the semantic type for "*king*" is person. For "*what <verb>*" questions the answer type is the type of the object of the verb. "*What is*" questions, which expect a definition as an answer ("*what is narcolepsy*?", "*what is molybdenum*?") are dealt specially. The answer type is the answer itself (a disease, a metal). However, it is often not possible to just look up the semantic type of the word, because lack of context does not allow identifying the right sense. Therefore, we treat definition questions as type-less questions: entities of any type are accepted as answers, provided they appear as subject in a TYPE- relation fact.

3.2. Linguistic Component

The Linguistic Component task is to map the NL input query to the linguistic-triple which is the subject, relation, object triple. After the execution of the linguistic component a set of syntactic annotations associated with the input query are returned. These annotations include information about the sentences, subjects, objects, adjectives and verbs. The output of this module is the logic representation of the query. Currently, the Linguistic component identifies 20 different linguistic categories or intermediate representations, including: basic queries requiring an affirmation/negation; or the big set of queries constituted by a W/H questions. The complete parse tree for the QASYO is shown in Figure 2.

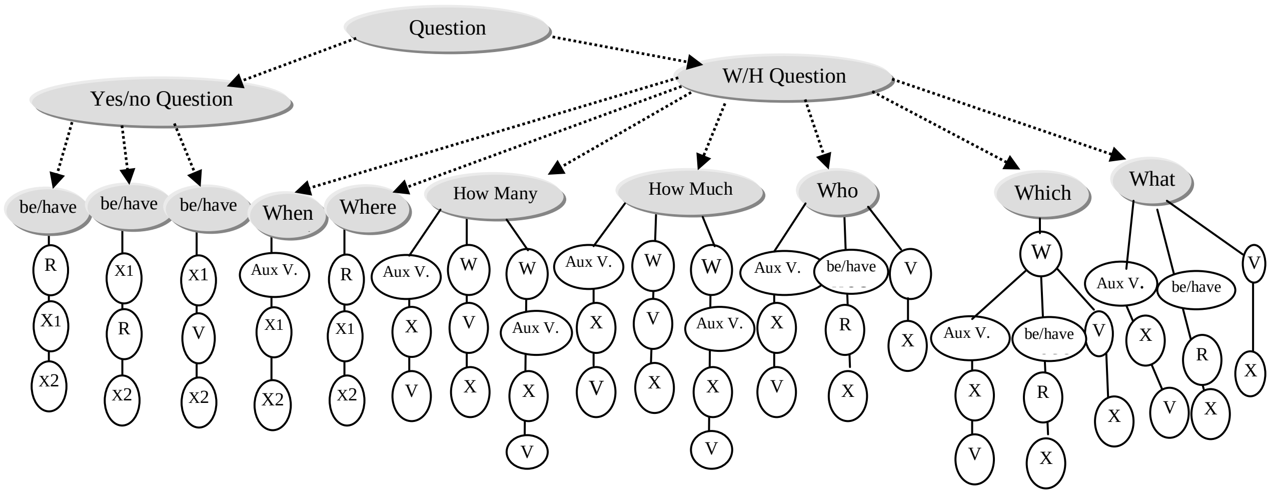

Figure 2. QASYO Parse Tree.

R is vector whose elements are words in the WR table which represent the nouns in the relation names such as budget, motto...etc used by YAGO. An example of the WR table is shown in Table 5. Since the universe of discourse we are working with is determined by and limited to the particular ontology used, there will normally be a number of discrepancies between the natural language questions prompted by the user and the set of terms recognized in the ontology. External resources like WordNet which is QASYO's primary lexical resource generally help in making sense of unknown terms, giving a set of synonyms and semantically related words which could be detected in the knowledge base. For example if the WR table contains the word "motto" and the user uses the word "slogan" in his question , the system will still be able to recognize the question. The size of the WR table grows linearly with the number of relation supported by YAGO. The version of YAGO ontology version 2008-W40-2 used in this research.

Table 5. Example of the WR Table

ID	Word	Related Relation
1	Currency	HASCURRENCY
2	motto	HASMOTTO

W is a vector whose elements are words used to generate the Query such as scientist, country, ..etc. The size of the W vector grows linearly with the number of relations supported by YAGO.

V is a vector whose elements are verbs in the verb table which represent verbs in the relation names such as died, happen...etc. An example of verb table is shown in Table 6. The same rules that are applies on R can be applied on V.

Table 6. Example of the Verb Table

ID	Verb	Related Relation
1	Act	ACTEDIN
2	Die	DIEDONDATE

X is any of the two arguments of a fact in the facts table. It is also the intended S/O in the query generator algorithm.

To be able to get the answer of the user question, the system should identify the linguistic category for which the user question belongs to. Depending on that, the system should detect at least two components of the query triple (i.e. the relation & one of the two arguments). These components will be detected if the system identifies the X element and the R or the V elements in the question. When the relation in the fact is known there are four cases to consider as shown in Table 7.

Table 7. Fact Cases

ARG1	ARG2	Case
known	known	This is a known fact [used only in YES/NO questions]
unknown	known	Valid question
known	unknown	Valid question
unknown	unknown	Invalid question

Some approaches expand one query to several queries, for example, by adding synonyms, then either concatenate these queries together with operator “OR” or keep them as separate queries. These approaches can increase the possibility of getting the correct answers, however, may also make the search more complicated and significantly prolong the search response time, and overall may be detrimental to the QA outcomes.

3.2.1. Relation Detection

1. Using the parse tree split the NL question into words and identify the list of candidate words R (and W if applicable), and a list of candidate verbs V. generate all possible consecutive word combination for R, W, and V with higher priority given to the longest word combination
2. Select related relation from WR table where word= [a candidate word combination]
3. Select related relation from Verb table where verb= [a candidate word combination].
4. Repeat steps 2 & 3 with another candidate combinations until a valid relation is detected.
5. If one or two relations are detected (they may be two relations if each of the two select statements in steps 2 and 3 return a valid relation) return relation(s) and continue;
   Else repeat step 2 & 3 but with WordNet synonyms of the candidate word combination.
6. If one or two relations are detected continue;
   else unrecognized question then fail.

In case of detecting two relations, the reliance will be on the detected X element. After detecting the X element, the system will be able to detect the intended relation by using the Domain-Range-Table (DRT). An example 0f DRT is shown in Table 8.

Table 8. Example of DRT

Domain	Relation	Range
Person	Born in	Location

3.2.2. Subject/Object (S/O) Detection

1. Split the NL question into words and identify the list of candidate words for X. The S/O candidates are generated from all the possible consecutive combination of words for X with higher priority given to the longest candidate.
2. Apply the select statement:
   SELECT ARG2 from Facts where relation="MEANS" and ARG1= “an S/O candidate”.

```
3. Repeat step 2 with another candidate until a valid S/O list is detected
4. If an S/O list is detected continue
   else unrecognized question then fail
```

In some relations such as "INFLUENCES", "HASACADEMICADVISOR", "HASCHILD", "ISMARRIEDTO" both the domain and range of these relations is a person so the query generated from this situation can be bidirectional, i. e.

```
      X  -> relation   ?
      ?  -> relation   X
```

The semantics of the question determines the appropriate direction to use in the query.

3.3. Query Generator

The Query-Triple is only a formal, simplified way of representing the NL-query, which we use mainly because at this stage we do not have to worry about getting the representation right in respect to the specific domain knowledge. The role of the intermediate representation is simply to provide an easy way to manipulate input for the Query generator phase. This design choice ensures the easy portability of the system with respect to both ontologies and natural languages

```
Input: the retrieved relation and S/O list
Output: ARG1 or ARG2
   1. use the DRT (domain range table ) to know the domain and
      range of the retrieved relation
   2. use the retrieved domain and range to specify the intended
      S/O [by using "TYPE" relation]
   3. by using the relation and S/O generate the query triple
      and get the list of results
```

Generating the query triple can be formulated as the following:

```
SELECT [ARG1 or ARG2] from FACTS where RELATIONS="retrieved
relation" AND [ARG2 or ARG1=" the intended S/O"]
```

3.4. Query Simplification Techniques

The computing speed has been considered a key issue in the development of QASYO. For a scalable Web-based QA system, trade-offs need to be made between speed and recall of answers. Thus QASYO does not try to find the best query; instead, it tries to locate the good enough query that will conduct the search task very fast. The focus has been laid on generating one simple query instead of an expanded one. Figure 3 presents three examples of questions handled by the QASYO system.

```
Example 1:
1. NL question
     What is the official currency of Brazil the big country?
2. Simplified question as recognized by the system
     What is the currency of Brazil?
3. Simplified question with the ontology concepts
     [Country] -> [HasCurrency] -> [???].
4. General form of the SELECT statement to get the answer
   SELECT ARG2 from FACTS where RELATION=[HasCurrency] AND ARG1=[country].
5. Incorporating the original NL question entities into the SELECT statement
   SELECT ARG2 from FACTS where RELATION=[HasCurrency] AND ARG1=[Brazil].
```

```
Example 2:
1. NL question
   Which scientist works at University of Cambridge the most important university
   in UK?
2. Simplified question as recognized by the system
     Which scientist works at University of Cambridge?
3. Simplified question with the ontology concepts
     [???] -> [worksAt] -> [University].
4. General form of the SELECT statement to get the answer
   SELECT ARG1 from FACTS where RELATION=[worksAt] AND ARG2=[University].
5. Incorporating the original NL question entities into the SELECT statement
   SELECT   ARG1   from   FACTS   where   RELATION=[worksAt]   AND   ARG2=[Cambridge
    University].
```

```
Example 3:
1.  NL question
    Where was Max Planck the great scientist and famous German person born?
2.  Simplified question as recognized by the system
    Where was Max Planck born?
3.  Simplified question with the ontology concepts
    [Person] -> [bornIn] -> [???].
4.  General form of the SELECT statement to get the answer
    SELECT ARG2 from FACTS where RELATION=[bornIn] AND ARG1=[Person].
5.  Incorporating the original NL question entities into the SELECT statement
    SELECT ARG2 from FACTS where RELATION=[bornIn] AND ARG1=[Max Planck].
```

Figure 3. Examples of the QASYO Process.

Several approaches are combined to form queries, including functional/special words deletion and word form modification.

3.4.1. Functional/Special Words Deletion

QASYO considers the adjectives, adverbs, nouns and verbs in the question. The deletion of functional words such as prepositions, determiners/ pronouns, and conjunctions can significantly shorten the size of the query. This can be used as a baseline of search engine specific query formation. Some words determined experimentally are not functional words but are acting as functional words either logically or structurally for example the words, "*kind of*" "*type*", "*sort*", and "*name*" can also be excluded from the query.

3.4.2. Word Form Modification

Some words in the original question are converted to another form before they are put in the query. Usually they are verbs, for example,

“*ended*” *in* question is converted to “end”
“*has*” in question is converted to “have”
“*acting*” in question is converted to “act”
“*wrote*” in question is converted to “write”

4. Experimental Results

QASYO allows a user who has a question in mind access the world’s knowledge simply by asking for the information they need in a way that is completely natural to them. The primary goal is to power a new kind of search experience which does not require users to learn specialized vocabularies, or to know the structure of the knowledge base. In the QASYO implementation, YAGO ontology version 2008-W40-2 was utilized and was converted to an oracle database. Visual C# was used for developing the system. We downloaded WordNet 2.1 and used wordnet.net [19, 25] which is a .Net Framework library for WordNet to be able to use the WordNet facilities in the system.

The performance of QASYO was compared to the performance of True knowledge a similar QA system. True Knowledge is an internet-scale answering engine that aimed to directly answer questions posed in plain English text, which is accomplished using a database of discrete facts [26]. The True Knowledge Answer Engine utilize a unique knowledge based and semantic technology that attempts to comprehend posed questions by disambiguating from all possible meanings of the words in the question to find the most likely meaning of the question being asked. It does this by drawing upon its database of knowledge of discrete facts. As these facts are stored in a form that the computer can understand, the answer engine attempts to produce an answer to what it comprehends to be the question by logically deducing from them. True Knowledge gathers information for its database in two ways: importing it from "credible" external databases which for them includes Wikipedia and from user submission following a consistent format and detailed process for input. As of August 18, 2010 the True Knowledge database overall contained 283,511,156 facts about 9,237,091 things. Thus, we asked 10 members none of whom had been involved in the QASYO project, to generate questions to evaluate QASYO's performance and to compare its performance to that of true knowledge. Because one of the aims of the experiment was to measure the linguistic coverage of the system with respect to user needs, we did not give them much information about the linguistic ability of the system. The questions were collected on the basis that both QASYO and true knowledge "know" the answer to (They have facts about the answers). We generated in total 100 questions, 91 of which were handled correctly by QASYO, versus 40 for handled correctly by true knowledge. This was a very good result, considering that no linguistic restrictions were imposed on the questions. Table 9 presents some sample assessment questions used for the QASYO evaluation.

5. Conclusions and Future Work

In this paper we have described the QASYO ontology-driven query answering system in the context of the semantic web scenario. QASYO presents an elegant solution in which different strategies are combined together to make sense of an NL query with respect to the universe of discourse covered by the ontology. Its ontology capabilities make QASYO a suitable NL front-end for the semantic web. The Future version of QASYO requires both an evaluation of its query answering ability as well as an evaluation of the overall user experience. Another extension would be to provide information about the nature and complexity of the possible changes required for the ontology and the linguistic component.

Table 9. Sample Assessment Questions

Question	QASYO	True Knowledge
What took place in the very famous country of France?	A	X
What is the capital of Egypt?	A	A
What is the political capital of Egypt?	A	X
What is the primary currency of the UK?	A	X
What did the writer Naguib Mahfouz write?	A	A
What did the Nobel prize winner Naguib Mahfouz write?	A	X
What is the famous motto of Honda?	A	X
Who works at Harvard?	A	A
Which scientist works at Harvard?	A	X
Who influenced the great scientist Albert Einstein?	A	X
Where did Adolph Hitler die?	A	A
Where did Adolph Hitler pass away?	A	X
When did the former American president Abraham Lincoln die?	A	X
What was located in Germany?	A	A
What was located in Germany the European country?	X	X

The letter A refers to "Answer" The letter X refers to "Doesn't Answer"

Acknowledgement

The authors would like to thank Ahmed M. El-hadidy, Ahmed Kouta, Omar Ibrahim and Mohammed Shalaby for their valuable help and cooperation in the QASYO project.

References

[1] Z. Zheng, “AnswerBus Question Answering System,” In Proceedings of Second International Conference on Human Language Technology Research, San Francisco, CA, USA, 2002.

[2] C. T. Kwok, O. Etzioni, and D. S. Weld, “Scaling Question Answering to the Web,” In Proceedings Tenth World Wide Web Conference, Hong Kong, China, 2001.

[3] S. Harabagiu, M. Pasca, and S. Maiorano, “Experiments with open-domain textual question answering,” In Proceedings of 18th International Conference on Computational Linguistics, 2000.

[4] J. Budzik and K. J. Hammond, “Learning for Question Answering and Text Classification: Integrating Knowledge-Based and Statistical Techniques,” In Proceedings of AAAI Workshop on Text Classification. Menlo Park, CA, 1998.

[5] P. Clark, J. Thompson, and B. Porter, “A knowledge-based approach to question answering,” In Proceedings of AAAI’99 Fall Symposium, Orlando, Florida. 1999.

[6] Askjeeves:http://askjeeves.com/2000

[7] F. M. Suchanek, G. Kasneci, and G. Weikum, “Yago: A Large Ontology from Wikipedia and WordNet,” MPI–I–2007–5-003, December 2007.

[8] F. M. Suchanek, G. Kasneci and G.Weikum, “YAGO: A Core of Semantic Knowledge Unifying WordNet and Wikipedia,” In Proceedings of 16th International World Wide Web Conference (IW3C2), pp. 697-706, 2007.

[9] M. Vergas-Vera and E. Motta, “AQUA ontology-based question answering system,” Lecture Notes in Computer Science, Vol. 2972. Springer-Verlag, Berlin, pp. 468–477, 2004.

[10] G. Attardi, A. Cisternino, F. Formica, M. Simi,and A. Tommasi, “PiQASso: Pisa Question Answering System,” In Proceedings of Tenth Text Retrieval Conference (TREC 2001), 2001.

[11] V. Lopez, M. Pasin, and Enrico Motta, “AquaLog: An Ontology-Portable Question Answering System for the Semantic Web,” Lecture Notes in Computer Science, Vol. 3532, Springer, Berlin, pp. 546-562, 2005.

[12] V. Lopez, and E. Motta, “Ontology-Driven Question Answering in AquaLog,” Lecture Notes in Computer Science, Vol. 3136. Springer-Verlag, Berlin, pp. 89–102, 2004.

[13] Bernstein, A., Kaufmann, E., G○hring, A., & Kiefer, C. (2005) , “ Querying ontologies: A controlled English interface for end-users,” In Y. Gil, E. Motta, V. R. Benjamins, & M. A. Musen (Eds.), International

semantic web conference. Lecture notes in computer science, Vol. 3729, Springer, Berlin, pp. 112–126. , 2005.

[14] W. Hunt, L. Lita, and E. Nyberg, "Gazetteers, WordNet, Encyclopedias, and the web: Analyzing question answering resources," Technical Report CMU-LTI-04-188, Language Technologies Institute, Carnegie Mellon, 2004.

[15] G. Ifrim and G. Weikum, "Transductive learning for text classification using explicit knowledge models," In Proceedings of PKDD, 2006.

[16] A. Ren, X. Du, and P. Wang, "Ontology-based Categorization of Web Search Results Using YAGO," In proceeding of 2009 International Joint Conference on Computational Sciences and Optimization, pp. 800-804, 2009.

[17] E. Kaufmann, A. Bernstein, and R. Zumstein., "Querix: A natural language interface to query ontologies based on clarification dialogs," In proceeding 5th International Semantic Web Conference (ISWC 2006), pp 980–981, 2006.

[18] D. M. Gerard, F. M. Suchanek, and A. Pease, " Integrating YAGO into the Suggested Upper Merged Ontology," In proceeding of 20th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2008), 2008.

[19] WordNet, http://wordnet.princeton.edu.

[20] C. Fellbaum, "WordNet: An Electronic Lexical Database," MIT Press, cogsci.princeton.edu/, 2000.

[21] RDF: http://www.w3.org/RDF/

[22] D. Mc Guinness and F. Van Harmelen, "OWL Web Ontology Language Overview," W3C Recommendation 10 (2004). http://www.w3.org/TR/owl-features/

[23] C. Matuszek, J. Cabral, M. Witbrock, and J. DeOliveira, "An introduction to the syntax and content of Cyc," In proceeding of AAAI Spring Symposium, 2006.

[24] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, R. Cyganiak, and Z. G. Ives, "Dbpedia: A nucleus for a web of open data," Lecture Notes in Computer Science, Vol. 4825, Springer, Berlin, pp. 722-735, 2007.

[25] http://opensource.ebswift.com/WordNet.Net/

[26] http://www.trueknowledge.com/