A Portable Natural Language Interface to Arabic Ontologies

Aimad Hakkoum[1], Hamza Kharrazi[2], Said Raghay[3]
Faculty of Science and Techniques, Cadi Ayyad University, Marrakesh, Morocco

***Abstract*—With the growing expansion of the semantic web and its applications, providing natural language interfaces (NLI) to end-users becomes essential to querying RDF stores and ontologies, using simple questions expressed in natural language. Existing NLIs work mostly with the English language. There are very few attempts to develop systems supporting the Arabic language. In this paper, we propose a portable NLI to Arabic ontologies; it will transform the user's query expressed in Arabic into formal language query. The proposed system starts by a preparation phase that creates a gazetteer from the given ontology. The issued query is then processed using natural language processing (NLP) techniques to extract keywords. These keywords are mapped to the ontology entities, then a valid SPARQL query is generated based on the ontology definition and the reasoning capabilities of the Web Ontology Language (OWL). To evaluate our tool we used two different Arabic ontologies: a Qur'anic ontology and an Arabic sample of Mooney Geography dataset. The proposed system achieved 64% recall and 76% precision.**

Keywords—Natural language interface; ontology; Semantic web; Arabic natural language processing (NLP)

I. Introduction

The semantic web is the natural extension of the current web, it is centered on enabling machines to understand web content so it can be easier for agents to look for information in a more precise and efficient way [1]. To accomplish this task, the semantic web proposed a set of new technologies; the most important one is the use of ontologies. An ontology can be defined as "an explicit specification of a conceptualization" [2]. Ontologies explicitly structure and represent domain knowledge in a machine-readable format so they can be incorporated into computer-based applications to facilitate automatic annotation of web resources, reasoning task and decision support [3].

Traditional search engines rely only on keyword search; they return a set of documents that contain one or more words of the initial query. On the other hand, semantic search engines rely on understanding the meaning of the user query through the use of NLP techniques and ontologies, then returning the exact answer from multiple data sources using semantic web technologies[4].

Many studies have shown the effectiveness of semantic search engines over classic keyword search engines when dealing with natural language queries. Singh [5] compared keyword search engines like Google and Yahoo to semantic search engines like Hakia and DuckDuckGo and concluded that semantic search returns more relevant answers.

In order to develop an ontology-based search engine, we have to create a natural language interfaces (NLI) to hide the complexity of the ontology to the end-user [6]. It will transform the user query expressed in natural language to a formal language query.

The aim of our research is to develop a portable NLI that can be used with any ontology or RDF store. The proposed system starts by a preparation phase that creates a gazetteer from the given ontology. When the user issues a query, it is processed using NLP techniques to extract keywords; these keywords are mapped to the ontology entities, then a valid SPARQL query is generated based on the ontology definition and the reasoning capabilities of the Web Ontology Language (OWL). Finally, the SPARQL query is executed against the ontology and the result is formatted and aggregated if needed before returning the answer to the user.

The rest of this article is organized as follow: Section 2 summarizes the related work. Section 3 presents the ontologies used to evaluate the system. Section 4 describes the proposed system. Section 5 discusses the evaluation of our system. Finally, Section 6 brings conclusions and sheds light on future work.

II. Related Work

There is a noticeable growth in using semantic web technologies for search systems development. This can be justified by the gain of accuracy using semantic search compared to keyword search as explained by Singh [5].

One way to take advantage of semantic web technologies is to utilize ontologies to expand the user query; this will improve the initial query by adding more related terms, and therefore improve the search results. This method has been adopted by many researchers, Alawajy [7] used domain ontologies and the Arabic WordNet (AWN) to provide reliable extended keywords in order to enhance Arabic web content retrieval. Besbes [8] proposed a new question analysis method based on ontologies, it consists of representing generic structures of questions by using typed attributed graphs and integrating domain ontologies and lexico-syntactic patterns for query reformulation.

Hattab [9] proposed the utilization of different levels of Arabic morphological knowledge to improve the search process in a search engine. The least degree of relationship is the strongest between the original word and the alternatives starting from the identical word, then its stem, its inflections and finally the root of the word.

These methods are suitable when fetching data from unformatted sources like text documents. We can benefit further from semantic web technologies by fetching data from RDF stores and knowledge bases. For that, we have to implement a NLI that will hide the complexity of the ontology to the end-user.

Based on different surveys ([10], [11]), most of the NLIs that were developed in the recent years are based on English. They can be classified into close domain NLIs and open domain NLIs. Close domain NLIs are adapted to a specific domain and therefore are more accurate and performant, on the other hand, open domain or portable NLIs are designed to work with any given ontology.

We are going to focus on portable NLIs, as it is the aim of our research. The first example is FREYA [12], it's an interactive NLI for querying ontologies. It uses syntactic parsing in combination with the ontology-based lookup in order to interpret the question, and involves the user if necessary. To help the user formulate his query, GINSENG [13] proposes to control user's input via a fixed vocabulary and predefined sentences structures through menu-based options, this approach gives a very good performance but cannot process all NL queries. Some NLIs like PowerAqua [14] designed a system that can query information from multiple ontologies, it gave promising results by obtaining a success rate of 70% correct answers of the evaluation questions.

Despite the fact that NLIs to ontologies have lately gained a considerable attention, existing approaches do not work with the Arabic language. The first research that worked in implementing a NLI using Arabic language is AlAgha [15] in 2015. This research proposed a system called AR2SPARQL; it translates Arabic questions into triples that are matched against RDF data to retrieve an answer. To overcome the limited support for Arabic NLP, the system does not make intensive use of sophisticated linguistic methods. Instead, it relies more on the knowledge defined in the ontology to capture the structure of Arabic questions and to construct an adequate RDF representation. The system achieved 61% average recall and 88.14% average precision.

In our previous paper [16], we proposed a semantic search system for the Qur'an. It is based on an Arabic NLI and a Qur'anic ontology that represents the Qur'an knowledge. Some of the algorithms used in this system are strongly dependent on the domain of the ontology and therefore cannot be applied to other domains. To overcome this limitation, we modified each algorithm to make it independent. Then we added more functionalities like approximate matching and user interaction in order to improve the performance and the accuracy of the system.

III. Used Ontologies

Recently, some efforts have been made to support Arabic in the semantic web. There are new Arabic ontologies that are being developed in different domains. The Islamic domain is one of the main topics of ontology development. The semantic Qur'an [17] created a multilingual RDF representation of the Qur'an structure, where the Qur'an ontology [16] extracted a set of concepts from the Qur'an like locations, living creations and events. Another interesting research is the translation of DBpedia to Arabic [18].

We based our selection of the evaluation ontologies on the following criteria:

- The ontology contains enough data to formulate at least 50 different questions.
- All the entities of the ontology have labels in Arabic.
- The ontology is available in a valid RDF representation format.

We chose two ontologies that meet these criteria. The first one is the Qur'an ontology [16]. The second ontology is Mooney GeoQuery dataset that contains data about the geography of the United States.

A. *Qur'an Ontology*

The Qur'an ontology[1] aims to represent the knowledge contained in the Qur'an in the form of Ontology. It represents the following concepts: chapters, verses, words, pronouns, verse topics, locations, living Creations and events. Table I presents some statistics of the ontology:

TABLE I. Qur'an Ontology Statistics

Object type	**Count**
Classes	49
Object properties	47
Data properties	23
Chapter	114
Verse	6236
Topic	1181
Living Creation	234
Location	69
Events	219

B. *Geography Dataset*

The Mooney GeoQuery dataset[2] describes the geography of the United States. Several English NLIs like FREYA and GINSENG used it to evaluate their system. It was translated to Arabic by AlAgha [15] for his system's evaluation. He translated all the classes and properties of the ontology, but not only 81 entities. We translated more labels to obtain 713 entities translated in Arabic. Table II shows some statistics of the ontology:

TABLE II. GeoQuery Ontology statistics

Object type	**Count**
Classes	9
Object properties	17
Data properties	11
State	51
Capital	51
City	351
Mountain	50
Road	40

[1] www.quranontology.com
[2] http://www.cs.utexas.edu/users/ml/geo.html

IV. NLI SYSTEM ARCHITECTURE

The NLI we are proposing is inspired from our semantic search system for the Qur'an [16]. After adapting the different algorithms of the system to be generic, we added a new component that enables the users to import new ontologies. Then we created a prototype and we run a set of tests using two different ontologies to analyze the possible causes of failure; this allowed us to improve the algorithms of matching, mapping and answer generation.

The structure of the proposed system is composed of five main components:

1) *Knowledge base preparation.*
2) *Query processing.*
3) *Entity mapping.*
4) *Formal query generation*
5) *Answer generation.*

A. *Ontology Preparation*

The ontology preparation component is triggered when the user imports a new ontology. It is composed of three sub-tasks: the extraction of the ontology definition, the generation of the inferred model and the construction of the gazetteer.

1) *Extraction of the ontology definition*

The extraction of the ontology definition consists of getting the ontology class hierarchy and the properties domain and range. Properties can be either object properties or datatype properties. Object properties have a domain and a range as a class; it is a relation between two entities. Datatype properties have a domain as a class and a range as a literal data like strings and numbers. The ontology definition will be used later in the automatic disambiguation task and the validation of the generated query triples.

2) *Inferred model generation*

The ontology specifies a set of facts and axioms. They can be used to generate new inferred triples in order to obtain an extended model. To avoid generating this model at each query execution, we will generate it the first time when the ontology is loaded and save it along with the ontology. We will use Jena Framework (http://jena.apache.org/) to manipulate the ontology model and generate the inferred model. Jena provides an API that enables to work with ontologies in different formats. It is widely used with semantic web technologies and is well documented and maintained.

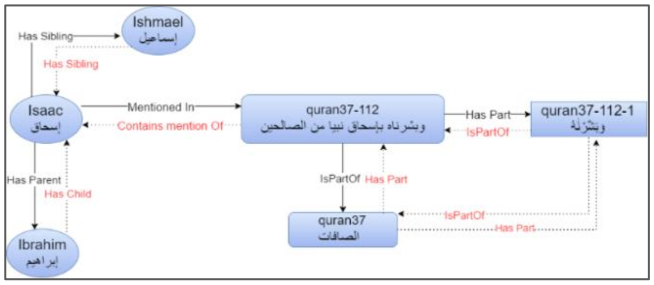


Fig.1. Example of inferred triples.

The inferred model will contain all the ontology triples in addition of the new deducted triples. Fig. 1 shows an example of the triples that we can get by this process. The black links represent the initial model, while the red links are obtained after generating the inferred model.

We will use this model to execute the SPARQL query generated by our system; this will increase the possibility to find an answer in the ontology. Table III shows the number of triples for the original model and the inferred model of the two evaluation ontologies:

TABLE III. INFERRED MODEL STATICTICS

	Quran	**Geo**
Initial model triples count	182 908	4 981
Inferred model triples count	473 353	14 830

3) *Gazetteer construction*

The construction of the gazetteer starts by the extraction of all the entities' labels. This includes the classes, the properties and the individuals. The extracted list of labels is then enhanced by generating synonyms from dictionaries and linguistic resources. The final step is to process each term of the gazetteer using NLP techniques.

a) *Synonyms generation*

Despite the recent efforts to support the Arabic language on the semantic web and NLP, it still lacks proper resources like WordNet and offline dictionaries. The Arabic version of the WordNet (AWN) developed by Abouenour [19] includes in its latest version about 17,785 words, it is still a work in progress and need more work to be comparable to the English WordNet which contains about 117,000 words. Therefore, using only the AWN will not be enough to generate synonyms. The other resource that we are going to use to accomplish this task is the on-line dictionary Almaany (www.almaany.com), it is the best tool we found and besides, no useable offline dictionary could be found.

b) *Linguistic processing*

The challenges of Arabic NLP are discussed in [20]. The main challenges are as follows:

- Lack of dedicated letters to represent short vowels, they are represented by diacritics.
- Changes in the form of the letter depending on its place in the word.
- Word agglutination: Arabic words often contain affixes representing various parts of speech. For example, a verb may embed within itself its subject and its object as well as the gender, person, number, and voice.

The first solution we are going to use to address these challenges is normalization. It consists of representing the Arabic text in a canonical form and thus avoiding the use of different forms to designate the same letter. The process of normalization is performed with Lucene Arabic analyzer[3]. An

[3] http://lucene.apache.org/

example of transformation is to replace all the forms of “alif” ("أ" ,"إ" ,"آ" and "ئ") by “ا”.

The second solution is stemming, which is the process of extracting the stem and the root by removing prefixes and affixes from words. There are several tools available for Arabic stemming. One of the recent and advanced tools is the Arabic Toolkit Service ATKS[4]. It contains multiple components like the Arabic parser, the Arabic speller and the morphological analyzer (SARF). These components are integrated into several Microsoft services such as Office, Bing, SharePoint and Windows. We are going to use SARF for stemming; in addition to the word root, it gives the stem, the morphological pattern and all the inflections of the word.

B. *Query Processing*

The user is assisted when entering his query with an autocomplete component. We use the terms of the gazetteer to give suggestions to complete each word of the query when the user enters two letters. After a new word is entered, the ATKS spell checker highlights the misspelled words.

Once the user validates his query, the query-processing component will start by tokenizing the query words, then removing stop words. We provide an initial list of stop words that the user can change depending on the domain of the ontology. After removing irrelevant words, we generate synonyms for each word using the AWN and Almaany. The final step is to process these words the same way the gazetteer terms were processed, which includes normalizing and stemming each word along with its synonyms.

C. *Entity Mapping*

The entity mapper is a critical component of any NLI system. It is responsible of mapping the query words to the ontology entities. To accomplish this task, we will start by comparing the query words to the gazetteer terms, if we find more than one match, we will try to choose one with an automatic disambiguation algorithm using the ontology definition. As a final step, we will ask the user to clarify the ambiguity manually by choosing one of the matching entities.

1) *String matching*

In order to match the user query with the ontology entities, we are going to combine two approaches of string matching: exact and approximate matching. We will start the comparison process by generating all possible n-grams starting from the highest n-gram that contains all the user query keywords to the unigrams that contain one word at a time. These n-grams are compared to the gazetteer terms according to the following order: 1) complete word; 2) normalized word; 3) word stem, 4) synonym; 5) normalized synonym; 6) synonym stem; 7) word root; 8) synonym root. We loop through all the possible n-grams starting from the highest ones, each time we found a match; we remove the n-gram words from the list of words to match. The matching algorithm is finished when we match all the words or when we arrive to unigrams.

The exact matching may not yield a result because of typographical errors, or phonetic similarity errors. In this case, we will use approximate matching. There are many methods to perform approximate matching [21], the most used one is known as the Levenshtein distance (also called “edit distance”). It consists of computing the minimum number of single characters edits needed to change one word into the other. We compute the Levenshtein distance on normalized words in order to have a result that is more pertinent. The similarity rate is computed as follows:

$$\boldsymbol{Sim}(Word1, Word2) = 1 - \frac{\boldsymbol{Lev}(Word1, Word2)}{\boldsymbol{Max}(Length(Word1), Length(Word2))}$$

Where **Lev**(Word1,Word2) is the Levenshtein distance

We compute this distance for all the gazetteer terms. We consider as a match the entity with the highest similarity rate. If this rate is over 90%, it is considered automatically as a match. Otherwise, we ask the user to validate the matching entity manually if this rate is between 90% and 70%. The user can then confirm the match or exclude the word from the answer generation process.

The approximate matching is especially useful for comparing proper names that can have multiple forms of writing in Arabic and for which the generation of the root does not return any value. Table IV shows some results obtained using approximate matching:

TABLE IV. EXAMPLES OF APPROXIMATE MATCHING

User word	**Ontology word**	**Similarity rate**
زكرياء	زكريا	83%
الذريات	الذاريات	87%
كالفورنيا	كاليفورنيا	88%
سان فرانسسكو	سان فرانسيسكو	92%

2) *Entity disambiguation*

This step is optional; however, it is very important when working with large ontologies because the same word can be used to identify different entities. As an example from the Qur’an ontology, we can find that the name of a chapter is the same as the name of a prophet or a topic. In this case, we are going to use the ontology definition and inference to try to find the accurate entity.

We have two ways to achieve the disambiguation: by class or by property. The first type of disambiguation corresponds to the scenario when the query contains a class and an individual from this class. In this case, if we have an ambiguity on the individual, we choose the one that is an instance of the class. The second type of disambiguation is used when the query contains an individual and a property that has as domain or range the type of this individual. In this case, we choose the individual that corresponds to the definition of the property.

To illustrate each type of disambiguation, let us analyze these two questions:

[4] http://research.microsoft.com/en-us/projects/sarf

1) من هم أبناء ابراهيم عليه السلام ؟ (Who are the children of Abraham ?)

2) ما على الحدود مع ولاية ميشيغان؟ (What on the border with the state of Michigan?)

The first question is executed against the Qur'anic ontology and the second one against the Geography ontology. Fig. 2 and 3 describe the mapping and the disambiguation process for each question:

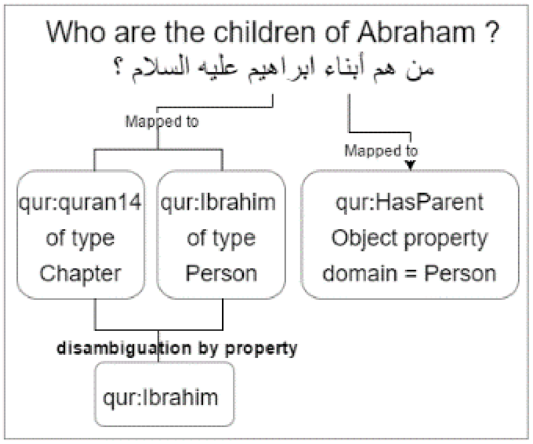

Who are the children of Abraham ?
من هم أبناء ابراهيم عليه السلام ؟
Mapped to
Mapped to
qur:quran14 of type Chapter
qur:Ibrahim of type Person
qur:HasParent Object property domain = Person
disambiguation by property
qur:Ibrahim

Fig.2. Automatic disambiguation of question 1.

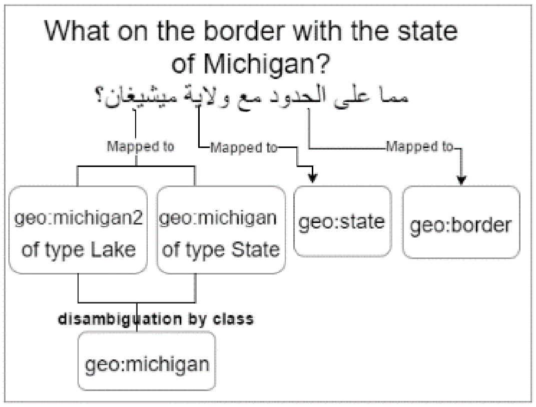

What on the border with the state of Michigan?
مما على الحدود مع ولاية ميشيغان؟
Mapped to
Mapped to
Mapped to
geo:michigan2 of type Lake
geo:michigan of type State
geo:state
geo:border
disambiguation by class
geo:michigan

Fig.3. Automatic disambiguation of question 2.

If the automatic disambiguation does not return a match, we ask the user through the interface to clear the ambiguity manually. We computed the number of each applied disambiguation when executing the evaluation questions on the system. Table V represents the list of the computed statistics:

TABLE V. DISAMBIGUATION STATISTICS

Dataset	**Quran**	**Geo**
Number of questions	60	90
Number of manual disambiguation	30	20
Number of auto disambiguation	11	10

We can see from these statistics that the contribution of the user is crucial in order to understand the meaning of the query. The Qur'an ontology needed more manual disambiguation; this can be explained by the fact that the same word is used to represent two or more concepts at the same time.

D. *Query Generation*

This component consists of generating a valid SPARQL query from the mapped entities. It is mainly based on the ontology definition and the reasoning capabilities of OWL.

1) Triple generation

The first step to construct the SPARQL query is to generate an initial set of query triples, these triples are in the following format: (s, p, o) where s, p and o respectively represent the subject, the predicate and the object. Each element can be either a known entity or a variable that the SPARQL query must return.

We loop through the mapped entities following the order of the user query, to each entity we apply a transformation function that will either create a new triple or modify an existing one. Fig. 4 describes the transformation algorithm of an entity (Ei):

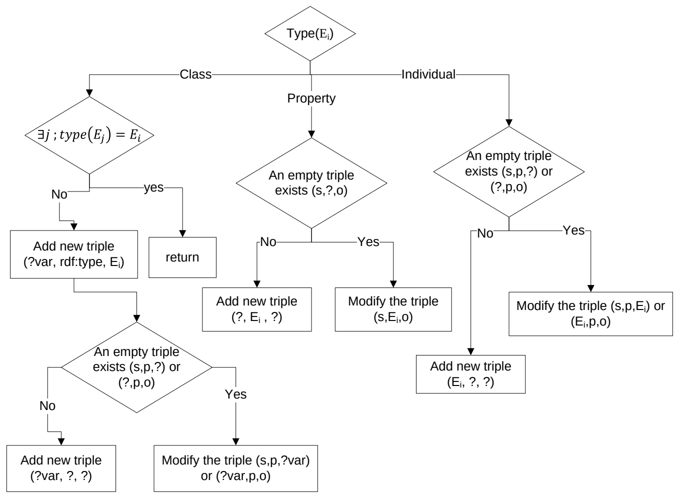

Type(Ei)
Class
Property
Individual
$\exists j\,; type(E_j) = E_i$
No
yes
Add new triple (?var, rdf:type, Ei)
return
An empty triple exists (s,p,?) or (?,p,o)
No
Yes
Add new triple (?var, ?, ?)
Modify the triple (s,p,?var) or (?var,p,o)
An empty triple exists (s,?,o)
No
Yes
Add new triple (?, Ei , ?)
Modify the triple (s,Ei,o)
An empty triple exists (s,p,?) or (?,p,o)
No
Yes
Add new triple (Ei, ?, ?)
Modify the triple (s,p,Ei) or (Ei,p,o)

Fig.4. Transformation function algorithm.

The first case of the algorithm is when the type of the entity is a class. If there is already an individual that have the type of this class, the entity will be ignored. Otherwise, we create a new triple (?var, rdf:type, Ei) and we add the variable into an incomplete triple (?var, p , o) or we create a new triple (?var, ? , ?).

Once we perform the transformation function on all entities, we obtain a list of triples: $(s_1, p_1, o_1), (s_2, p_2, o_2), \ldots$

These triples may be incoherent and not representing a valid RDF triple, therefore in the next step we will perform more processing to validate and expand these triples in order to obtain the final triples list.

2) Query expansion

A valid query triple (s, p, o) must verify the following integrity conditions:

$$\text{domain}\ (p) = \text{type}\ (s)\ \text{and range}\ (p) = \text{type}\ (o)$$

In order for each triple to comply with these conditions, we will apply a set of changes by expanding the triple or by simply changing the order between the object and the subject.

When the property is a variable, we check the ontology definition for a property that satisfies the integrity condition, if we do not find any, we expand the initial triple to obtain the two triples: (s, p', o'), (o', p'', o). This scenario corresponds to two entities on the ontology graph that are not linked directly by a relation so we have to add an intermediate node in order to link them.

When the property of the triple is not a variable and does not satisfy the integrity condition, we add a new individual (i) and a new property (p') to obtain two triples as a result. Table VI lists all the possible transformation of a query triple:

TABLE VI. QUERY TRANSFORMATION SCENARIOS

Condition	Transformation
domain(p) = s and range(p) ≠ o	(s, p, i) , (i, p', o)
domain(p) = o and range(p) = s	(o, p, s)
domain(p) = o and range(p) ≠ s	(o, p, i) , (i, p', s)
domain(p) ≠ s and range(p) = o	(s, p', i) , (i, p, o)
domain(p) ≠ o and range(p) = s	(o, p', i) , (i, p, s)

3) *SPARQL generation*

SPARQL is the W3C (World Wide Web Consortium) standard to query RDF data stores. A SPARQL query uses a "SELECT" statement to define which data the query should return, and a "WHERE" statement that defines a graph pattern where some nodes are known and others are not, the query should then find all possible subgraphs that meet the pattern.

The generation of the SPARQL query consists of putting the variables in the SELECT statement and the query triples in the WHERE statement. For each variable, we add an optional variable that corresponds to the Arabic label with an optional filter as follows:

$SELECT\ \ ?var1\ \ ?var1Label\ \ ?var2\ \ ?var2Label\ \ \ldots$

$Where\ \{\ (s_1, p_1, o_1).(s_2, p_2, o_2), \ldots\}$

$OPTIONAL\ \{?var1\ rdfs\!:\!label\ \ ?var1Label.$

$\quad Filter\ (\ langMatches(lang(?var1Label), \text{ar})).\}$

E. *Answer Generation*

The SPARQL query is executed against the inferred model of the ontology using the Jena Framework. After getting the result of the query, we apply two functions to format and remove redundant information. The first function removes duplicate rows and columns that contain the same data. The second function perform an aggregation on the result, it is used when the first column contains a limited number of non-duplicated values while the number of lines is very important. In this case, the rows are aggregated using the values of the first column, which is considered the main object of the question.

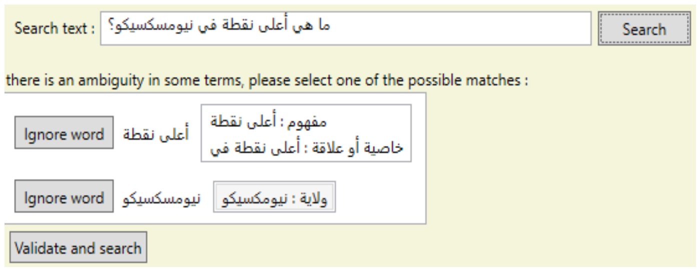
F. *System Interface*

We designed a simple interface that implements the NLI system. It consists of a desktop application. It does not need any installation and can run directly after downloading the binaries from the project webpage.

The application allows the user to perform a set of actions in addition to using the search engine. The application menu enables the user to import new ontologies and remove existing ones. When an ontology is selected, the user can use the menu to edit the gazetteer or the stop words list.

To use the search engine, the user must choose the desired ontology from the list of ontologies and enter his query in Arabic. The autocomplete component will propose suggestions when the user enters two letters for a word. When the user executes the search, he may need to disambiguate some of the query words. Here is an example of the disambiguation process:

Fig. 5 shows an example of the disambiguation process. In this example, we have one word that can be mapped to two entities from the ontology, and one word that was mapped using approximate matching. The user can select one entity for the first word, and validate the approximate matching for the second word. He can also exclude a word and it will be removed from the rest of the process.



Search text : ما هي أعلى نقطة في نيومسكسيكو؟

there is an ambiguity in some terms, please select one of the possible matches :

أعلى نقطة
مفهوم : أعلى نقطة
خاصية أو علاقة : أعلى نقطة في

نيومسكسيكو
ولاية : نيومكسيكو

Fig.5. Disambiguation process screen.

The search result is displayed in a group of tabs. The first one is the answer of the query; it may contain a single element, a list with a single column or a table with multiple columns. The other tabs describes the details of each component of the system, they can help the user understand how the answer was generated. The last tab contains the SPARQL query used to retrieve the answer. The user can modify it and click on the "Exec query" button, the result of the query is then displayed in the first tab.

V. SYSTEM EVALUATION

A. *Evaluation Questions*

We prepared a set of evaluation questions for the two ontologies. We ensured that the questions are in the scope of the ontology and the system has the ability to answer them, this will allow us to analyze the causes of failure for unanswered queries in order to improve the system effectiveness.

For the Qur'an ontology, we used the sample questions available on the project website; we added some more questions to reach 70 questions. For the Mooney GeoQuery dataset we took the 877 questions used by AlAgha [15] to evaluate his system, then we removed the questions that are rather similar. We also removed the improperly formatted ones to obtain 90 questions.

The evaluation questions along with the two ontologies can be accessed in the project webpage.

B. Evaluation Results

To calculate the performance of the proposed NLI system, we executed all the evaluation questions using the user interface after importing the two ontologies. We analyzed the result of each question in order to compute the statistics about the mapping and disambiguation components, we also classified the reasons behind the system's failure.

We are going to compute the following metrics: precision and recall. The precision is the number of correctly answered questions over the number of questions that the system provided an answer for, while the recall is the number of questions correctly answered over the number of all questions in the dataset.

The webpage of the project contains the detail of the evaluation process. For each question, we define if the system provides an answer or not, and if this answer is correct. Table VII shows the general result of the evaluation:

TABLE VII. EVALUATION RESULTS

Ontology	**Quran Ontology**	**Geography**	**Total**
Total number of questions	70	90	160
Questions with an answer	53	80	133
Questions with correct answer	47	56	103
Precision	88%	70%	76%
Recall	67%	62%	64%

The system gave answers to 133 questions, among which 103 are correct, thus achieving 64% average recall and 76% average precision. We can see that the Qur'an ontology gives more precision than the geography dataset. This can be explained by the fact that the Qur'an ontology defines several possible labels to identify its entities; this helps the mapping of the user keywords to the ontology entities.

The NLI (AR2SPARQL) proposed by [15] claims a recall of 61% and a precision of 88.14%. This system is not publically available; therefore, we cannot make a pertinent comparison of the performance of the two systems, because we have to use the same questions to evaluate both systems. On the overall, our system is based on the same approach of AR2SPARQL, which is the use of the knowledge defined in the ontology to process the user query. Both systems do not make intensive use of sophisticated linguistic and semantics techniques. However, some components are different between the two; for instance, we use approximate matching and user interaction for entity mapping. Our system also relies on the involvement of the user to clarify his question and the use of a set of rules to validate and enhance the SPARQL query.

C. System Failure Analysis

We analyzed the reasons of failure for each question that the system could not answer correctly, we classified these reasons into three categories: mapping error, complex questions and uncovered questions. The first category represents 40%; the second 50% while the last one 10%.

The mapping error category represents the questions where we could not map the user terms to the entities of the ontology or that we mapped a term to the wrong entity. This kind of error is due to the challenges of Arabic NLP discussed earlier. Some of the unanswered questions can be fixed by editing the gazetteer of the ontology via adding new alternative labels for the entities. Where some questions need to be reformulated by the user in order to get an answer.

The second category represents complex questions that require adding more rules and algorithms to the system in order to be able to answer them. We can identify different kinds of complex queries:

- Long questions with term dependencies: This kind of questions needs deep linguistic analysis to extract the dependencies between terms. Some of the English NLIs used syntactic parsing and part-of-speech tags to extract the parse tree of the question, this helps understand the question's structure and to generate valid query triples. The use of this method with the Arabic language is still challenging due to the lack of efficient NLP tools and the high productivity of the Arabic language.
- Questions with superlatives and comparatives: The interpretation of comparative and superlative words depend on the domain of the ontology, and even in the same ontology, we can find multiple interpretations for the same term.
- Vague questions: The user may use vague expressions to formulate his question, making the understanding of its meaning very difficult. An example of this type of question is "ما يمكنك ان تقول لي عن سكان ميسوري؟" (What can you tell me about Missouri residents?), we can see that the question is not precise enough because we have two properties that describe the population: the number of inhabitants and the population density.

The third category represents the questions that the ontology does not contain an answer for even if the query processing was successfully performed. This kind of questions needs the enrichment of the ontology by adding more properties and entities.

D. Publishing the Results

We created a webpage for the project at the following address: https://sites.google.com/site/arabicnlisystem. It contains all the resources necessary to use and evaluate the NLI system.

We also shared the source code of the project for other researchers to leverage on their research. The source code is composed of two layers. The first one is the user interface that contains the definition of the application forms, and the second one is the business layer that contains all the system's logic and algorithms.

VI. CONCLUSION

In this research, we developed a portable NLI to Arabic ontologies. The system can be used with any ontology that defines Arabic labels to its entities. We tested our approach on two different ontologies that represent two separate domains, the system gave better results when the ontology is well structured and provides alternative labels to describe its entities.

We used existing Arabic NLP tools to process the ontology labels and the user question, this allowed us to map the user terms to the ontology entities, and then we relied on the ontology definition and the reasoning capabilities of OWL to create a SPARQL query that will be used to extract the answer from the ontology.

We believe that our work will be a step toward adopting semantic search engines for the Arabic language. The researchers can integrate our system library in their projects with minimum effort to have a semantic search tool for their ontologies.

The next step of our research is to study the reasons behind the system's failure and try to improve the capabilities of the system to answer unmanaged questions patterns. Another perspective is to improve the search system to answer questions from multiple ontologies. This will require the use of ontology alignment frameworks in order to allow multiple ontologies to interoperate.

REFERENCES

[1] Y. Badr, R. Chbeir, A. Abraham, and A.-E. Hassanien, "EmergentWeb Intelligence: Advanced Semantic Technologies.," in Springer Science & Business Media, 2010, p. Page: 26-36.

[2] T. R. Gruber, "Toward principles for the design of ontologies used for knowledge sharing," Int. J. Hum. Comput. Stud., vol. 43, no. 5–6, pp. 907–928, 1995.

[3] N. Shadbolt, T. Berners-Lee, and W. Hall, "The Semantic Web - The Semantic Web Revisited," IEEE Intell. Syst., vol. 21, no. 3, p. 96, 2006.

[4] T. Tran, P. Haase, and R. Studer, "Semantic search--using graph-structured semantic models for supporting the search process," in International Conference on Conceptual Structures, 2009, pp. 48–65.

[5] J. Singh, "A Comprative Study Between Keyword and Semantic Based Search Engines," in International Conference on Cloud, Big Data and Trust, 2013, pp. 13–15.

[6] E. Kaufmann and A. Bernstein, "How useful are natural language interfaces to the semantic Web for casual end-users?," in The Semantic Web. Springer Berlin Heidelberg, 2007, vol. 4825 LNCS, pp. 281–294.

[7] A. M. Alawajy and J. Berri, "Combining semantic techniques to enhance arabic Web content retrieval," 2013 9th Int. Conf. Innov. Inf. Technol. IIT 2013, pp. 141–147, 2013.

[8] G. Besbes, H. Baazaoui-Zghal, and A. Moreno, "Ontology-based question analysis method," in International Conference on Flexible Query Answering Systems. Springer Berlin Heidelberg., 2013, pp. 100–111.

[9] M. Hattab, B. Haddad, M. Yaseen, A. Duraidi, and A. A. Shmais, "Addaall Arabic Search Engine: Improving Search based on Combination of Morphological Analysis and Generation Considering Semantic Patterns," in The second International Conference on Arabic Language Resources and Tools, Cairo, Egypt., 2009, pp. 159–162.

[10] S. Kalaivani and K. Duraiswamy, "Comparison of question answering systems based on ontology and semantic web in different environment," J. Comput. Sci., vol. 8, pp. 1407–1413, 2012.

[11] A. Bouziane, D. Bouchiha, N. Doumi, and M. Malki, "Question Answering Systems: Survey and Trends," Procedia Comput. Sci., vol. 73, no. Awict, pp. 366–375, 2015.

[12] D. Damljanovic, M. Agatonovic, and H. Cunningham, "FREyA : an Interactive Way of Querying Linked Data Using Natural Language," in Extended Semantic Web Conference. Springer Berlin Heidelberg., 2011, pp. 125–138.

[13] A. Bernstein, E. Kaufmann, and C. Kaiser, "Querying the Semantic Web with Ginseng : A Guided Input Natural Language Search Engine," in 15th Workshop on Information Technologies and Systems Las Vegas NV, 2005, no. December, pp. 45–50.

[14] V. Lopez, V. Uren, M. Sabou, E. Motta, and Acm, "Cross Ontology Query Answering on the Semantic Web: An Initial Evaluation," K-Cap'09 Proc. Fifth Int. Conf. Knowl. Capture, pp. 17–24, 2009.

[15] I. Alagha, "AR2SPARQL : An Arabic Natural Language Interface for the Semantic Web," in International Journal of Computer Applications (0975 –8887), 2015, vol. 125, no. 6, pp. 19–27.

[16] A. Hakkoum and S. Raghay, "Semantic Q&A System on the Qur'an," Arab. J. Sci. Eng., vol. 41, no. 12, 2016.

[17] M. A. Sherif and A. C. Ngonga Ngomo, "Semantic Quran A multilingual Resource for Natural-Labguage Processing," Semant. Web J., vol. 6, pp. 339–345, 2009.

[18] H. Al-Feel, "The Roadmap for the Arabic Chapter of DBpedia," Math. Comput. Methods Electr. Eng., pp. 115–125, 2015.

[19] L. Abouenour, K. Bouzoubaa, and P. Rosso, "On the evaluation and improvement of Arabic WordNet coverage and usability," Lang. Resour. Eval., vol. 47, no. 3, pp. 891–917, 2013.

[20] A. Farghaly and K. Shaalan, "Arabic natural language processing : Challenges and solutions," in ACM Transactions on Asian Language Information Processing (TALIP), 2009, vol. 8, no. 4, pp. 1–22.

[21] G. Navarro, "A guided tour to approximate string matching," ACM Comput. Surv., vol. 33, no. 1, pp. 31–88, 2001.