DBPEDIA BASED FACTOID QUESTION ANSWERING SYSTEM

Adel Tahri[1] and Okba Tibermacine[2]

[1]Department of Computer Science, Biskra University, Biskra, Algeria
adel.tahri@univ-biskra.dz
[2]Department of Computer Science, Biskra University, Biskra, Algeria
o.tibermacine@univ-biskra.dz

ABSTRACT

Question Answering (QA) is an area of natural language processing research aimed at providing human users with a convenient and natural interface for accessing information. Nowadays, the need to develop accurate systems gains more importance due to available structured knowledge-bases and the continuous demand to access information rapidly and efficiently. In This paper we propose a new architecture to develop a factoid question answering system based on the DBpedia ontology and the DBpedia extraction framework. We use the SVM learning machine algorithm to train the systems question classifier to achieve a high accuracy ratio. The design and implementation steps are covered with sufficient details. In addition, tests and experiment results of the developed system are presented with a short discussion about the systems efficiency.

KEYWORDS

Natural Language Processing, Question Answering, Ontology, Wikipedia, DBpedia, Machine Learning and SVM.

1. INTRODUCTION

Information Retrieval (IR) [1] is an open research field where scientists attempt to automatically extract information from large corpora. A well-known sample of IR systems is the search engines on the Web like Google, Yahoo!, etc. These systems allow the user to introduce a query like "Who was the first President of Algeria". Then, they answer the query by returning a list of indexed web pages or documents that hold the given list of words. But probably what the user like better is to get the answer explicitly. Thus, the system answers directly the previous query by returning "Ahmed Ben Bella". Question Answering (QA) [2] systems are dedicated to handle this task. As their name indicates, they are IR systems that answer questions.

Nowadays, many factors has reignited the interest in developing QA systems, including the booming increase of the web data, the improvements in information technology and the continuous demand of internet users to access to the correct information rapidly. So, the availability of huge document collections (e.g., the web itself), combined with improvements in information retrieval (IR) and Natural Language Processing (NLP) techniques, has attracted the development of a special class of QA systems that answers natural language questions by consulting documents on the web [3][4][5][6][7][8], or using special knowledge base such works presented in [9][10][11] and [12].

One of the richest corpora, available on the web, is the free encyclopaedia Wikipedia[3] that holds thousands of information in all knowledge fields. Wikipedia represent this information in articles

[3] The multilingual free-content encyclopaedia on internet : http://www.wikipedia.org

DOI : 10.5121/ijwest.2013.4303

under a semi-structured way. DBpedia[4] is a project that aims at extracting information based on the semi-structured data presented within the Wikipedia articles, interlinking it with other knowledge bases and publishing this information as Resource Description Framework (RDF) documents freely on the Web. Thus, it's crucial to use this rich source of structured information in building new QA systems.

Hence, we propose in this paper a factoid question answering system based on the DBpedia extraction framework and the DBpedia ontology. By factoid question answering system we mean a system that can understand and response simple question types. We coin the developed system by "Selni" which means ask me in the Arabic language. SELNI is a sentence level question-answering system that integrates natural language processing, ontologies, machine learning and information retrieval techniques.

The paper is organised as follow, in section two we present the DBpedia project. The architecture of the system is presented in section three. An implementation of the proposed architecture is covered in section four. Tuning, experiments and evaluation of the developed system is discussed in section five.

2. DBPEDIA

Wikipedia has grown into one of the central knowledge sources of mankind. Actually, it's the largest encyclopedia on the web maintained by thousands of contributors. It's being the 7th most visited website according to Alexa.com[5] (June 2013). Wikipedia is available in more than 285 languages, and the English Wikipedia contains more than 3.5 million articles. The DBpedia project [13] aims to extract information from Wikipedia and make this information available on the emerging web of data [14].

The three major parts of the DBpedia project are the website, the datasets and the Extraction Framework. The dataset based on English Wikipedia infoboxes is probably the most interesting one. The extraction work is performed by the DBpedia Extraction Framework. The DBpedia ontology[6] has been created for the purpose of classifying this extracted data. It's a cross-domain ontology based on infobox templates in Wikipedia articles. The ontology currently covers 359 classes which form a subsumption hierarchy and are described by 1,775 different properties.

In addition, the DBpedia 3.8 knowledge base describes [7]: 3.77 million things, out of which 2.35 million are classified in a consistent Ontology, including 764,000 persons, 573,000 places (including 387,000 populated places), 333,000 creative works (including 112,000 music albums, 72,000 films and 18,000 video games), 192,000 organizations (including 45,000 companies and 42,000 educational institutions), 202,000 species and 5,500 diseases.

The DBpedia knowledge base has several advantages over existing knowledge bases: It covers many domains, it represents real community agreement, it automatically evolves as Wikipedia changes, and it is truly multilingual.

3. SYSTEM ARCHITECTURE

We propose to build a question answering system that answers English questions based on the DBpedia Infobox ontology. Hence, the next sections describe the necessary steps to build such

[4] The DBpedia Project : http://www.dbpedia.org
[5] http://www.alexa.com/siteinfo/wikipedia.org
[6] http://dbpedia.org/Ontology
[7] http://blog.dbpedia.org/category/dataset-releases/

system. In fact, three steps are followed to build this system as depicted in Schema (Figure 1). The first step deals with the comprehension of the question and the detection of its answer type (the answer is human, numeric value, Location …). This later lay out on decision Question-Model. The decision question model is built based on question classification and machine learning techniques, as we'll explain it in subsection (3.1). In fact, these question models are built only once and reused every time the system is used. We label the first step by "**Question Classification and decision Model Generation**".

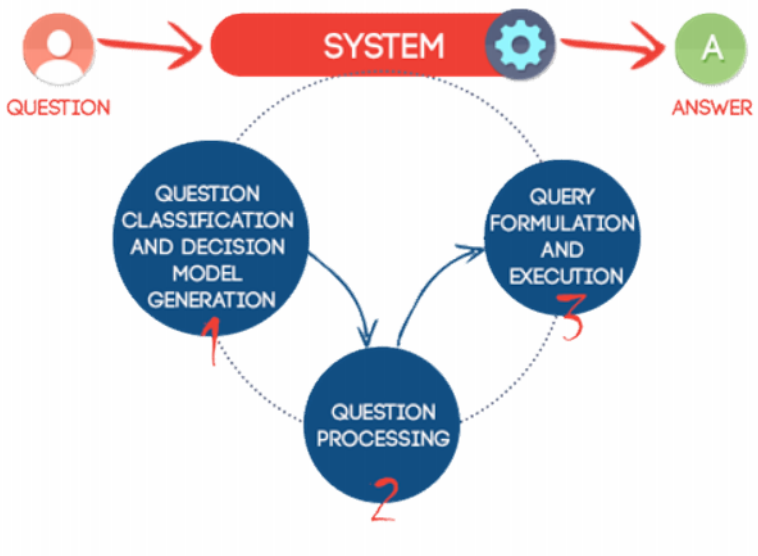

Figure 1. Global Question Answering System Architecture

The second step is "**Question Processing**", where we split the question into a set of words. These words are gathered to form question features. And features allow us based on Question-Model to extract Resources and Keywords which correspond to DBpedia Ontology classes. Full details about question processing are given in subsection (3.2).

In the third step, the resources and keywords extracted from the question are used to build SPARQL query and execute it by interrogating the DBpedia ontology. The result of the query is the answer of the given question. We label the third step by "**Query Formulation and Execution**" and full details about this step are given in subsection (3.3).

3.1 Questions Classification and Decision Model Generation

The nucleus of the proposed system is the decision model issue from machine learning computation; the decision model is built from question classification using the SVM (Support Vector Machines) algorithm with a dataset of 3000 question. The goal of this model is to return the answer type of a question, for example if the system gets the question "What is the capital of Norway?" the answer type of this question is "City" which belongs to the class "Location". Hence, the system should seek for a city as an answer to the given question.

The question classification and decision model generation are ensured by two components which are depicted in (figure 2). The first component is the question classifier which pre-processes the question dataset and trains it by the SVM algorithm. And the second component is the decision model generator which establishes the models that we will use in the proposed question

answering system. The mechanisms of both question classifier and decision model generator are explained in the subsections (3.1.1 and 3.1.2) respectively.

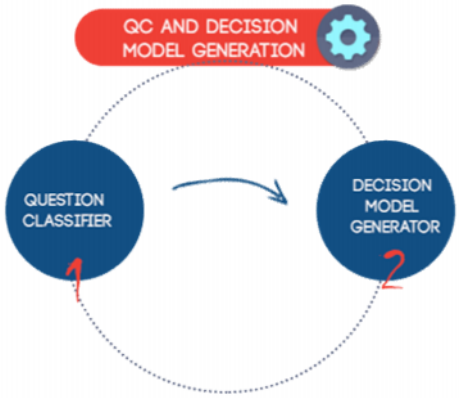

Figure 2 : Questions Classification and Decision Model Generation components

3.1.1. Question Classification

Question Classification (QC) is a vital component of any Question Answering system. Research on QA such like those done by Srihari and Li [15] has demonstrated the importance of classifying questions with respect to their answer types. Knowing the answer type, reduces processing and effort and provides a feasible way to select correct answers from among the possible answer candidates. In literature, Question Classification is defined as "given a question, map it to one of k classes, which provide a semantic constraint on the sought-after answer" [15]. Hence, the overall goal of question classification is to categorize questions into different semantic classes that impose constraints on potential answers.

The first question we have to answer is what technique or approach we've to use in our question classifier? In fact, many techniques and approaches have being used in question classification. Nevertheless, machine learning approaches (especially SVM) have proved to achieve the highest answer type accuracy according to the comparative study of Li and Roth which has the best performing question classifier yet [15]. For that reason, we use the Support Vector Machines algorithm to train the systems question classifier (subsection A). Also, we adopted the coarse-grained classes and the fine-grained classes of answers types proposed by the authors of the previous work. Where, they used 6 coarse-grained classes plus the Boolean class. And 50 fine-grained classes, all classes are listed in (Table 1).

Coarse classes	Fine classes
Human	description, group, individual, title
Location	city, country, mountain, other, state, continent, body of water, restaurant, retail, attraction
Entity	animal, body, color, creative material, currency, disease/medicine, event, food, instrument, language, letter, other, plant, product, religion, sport, substance, symbol, technology/method, terminology, vehicle, word, weather condition
Description	definition, description, manner, reason

Abbreviation	abbreviation, expansion
Number	code, count, date, distance, money, ordinal, other, percentage, period, speed, temperature, volume/size, weight
Boolean	Boolean

Table 1:Coarse-grained and Fin classes used by Li and Roth [15]

These classes denote the answer type of question; for example let's have the question "What is the capital of Norway?" the coarse-grained class for this question is LOC which means location. And the fine-grained class for this question is City. Other illustration samples are presented in (Table 2).

CLASS	DESCRIPTION	EXAMPLE
ENTY	entity	What is the largest snake in the world?
HUM	human	What actress has received the most Oscar nominations?
LOC	location	Where is the Orinoco?
DESC	description	What are equity securities?
ABBR	abbreviation	What does S.O.S stand for?
NUM	numeric	How much does a new railroad car cost?

Table 2: Question samples with their corresponding Coarse-grained

A) Support Vector Machines (SVMs) algorithm

Support Vector Machines (SVMs) [16], are supervised learning machines based on statistical learning theory that can be used for pattern recognition and regression. The SVM are linear functions of the form $f(x) = \mathbf{w} \bullet \mathbf{x} + b$, where $(\mathbf{w} \bullet \mathbf{x})$ is the inner product between the weight vector **w** and the input vector **x**. The SVM is a binary classifier by setting the class to 1 if $f(x) > 0$ and to -1 otherwise. SVMs are based on the class of hyperplanes

$$\mathbf{w} \bullet \mathbf{x} + b = 0, \mathbf{w} \in R^n, b \in R$$

Where, the vector **w** defines a direction perpendicular to a hyperplane while varying the value of **b** move the hyperplane parallel to itself. Which basically divide the input space into two: one part containing vectors of the class −1 and the other containing those that are part of class +1. If there exist such a hyperplane, the data is said to be linearly separable. To find the class of a particular vector x, we use the following decision function. For multi-class SVM, **Directed Acyclic Graph** [7] strategy has been demonstrated to work well.

- **Directed Acyclic Graph: A** Directed Acyclic Graph (DAG) is a graph whose edges have an orientation and no cycles. A Rooted DAG has a unique node such that it is the only node which has no arcs pointing into it. A Rooted Binary DAG has nodes which have either 0 or 2 arcs leaving them. We will use Rooted Binary DAGs in order to define a class of functions to be used in classification tasks. The class of functions computed by Rooted Binary DAGs is described in (Figure 3). Certainly, we could not cover all SVM concepts, hence for farther reading consult [8]

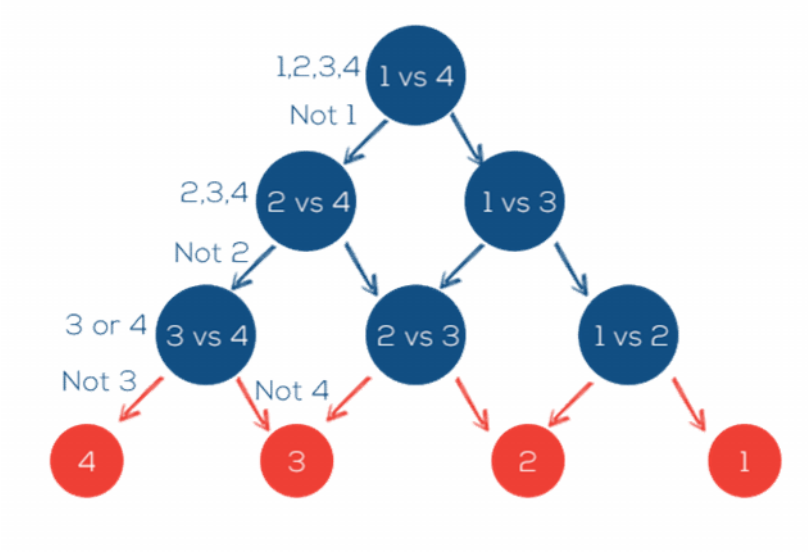

Figure 3 : Decision DAG for selecting the best class

B) Dataset

To train the question classifier we use a dataset; it consists of two sets of annotated questions. The first one contains 5.500 questions for the training task, and the second holds 500 questions for the testing task. The dataset is retrieved from TREC 10[8].

C) Question Classifier

Logically, the question classifier makes use of a sequence of two simple classifiers each utilizing the SVMs. The first classifier classifies questions into coarse classes (Coarse Classifier), and the second classifier classifies questions into fine classes (Fine Classifier). A feature extractor automatically extracts the same features for each classifier. The second classifier depends on the first one.

The goal of the question classifier is to generate a decision models that could be used to classify future question. More details on the work of the classifiers and how to generate decision models are given in the following subsection.

3.1.2. Decision Model

For both coarse and fine classifiers, the same decision model is used to choose class labels for a question. (Figure 4) show question classification steps followed to generate SVM models. These steps are:

1): The classifier starts by tokenizing questions of the dataset (Train and Test sets). Tokenization role is splitting up the entire questions to a bag of words. Which serves later to the construction of the question vector (called also sample vector). More details about tokenization are done in the subsection (A).

2): The classifier removes stop words from the bag of words generated after the tokenization process. Details about stop words removing are given in point (B).

[8] TREC : http://**trec**.nist.gov/

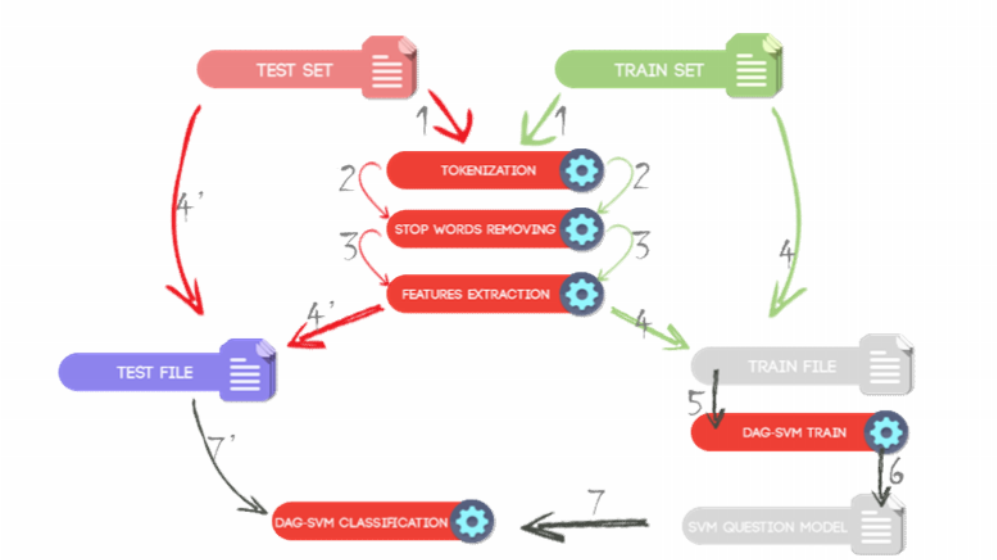

Figure 4 : Question Classification & model generation

3): The classifier uses the bag of words to extract features. We have a lot of kind of feature as explained in point (C). These features must be presented numerically as vectors. To do so, each word is indexed by an integer which represents also its index in the sample's (question) vector. The resulting data structure is a matrix with dimension [number of question, number of features]. We denote with X to the resulted matrix. The value X[n, m] is the frequency value of the feature m in the question n.

4): The classifier constructs a train file from the matrix and the sample vector.

4'): The classifier constructs a test file from the test Matrix and the sample vector.

5 & 6): Once the train file is ready for process, the classifier starts the training process (DAG-SVM) based on the linear SVM algorithm. After the training process is done the decision model (SVM Question model) is generated.

7 & 7'): After constructing the decision model, it's time to evaluate its accuracy. To do so, we use the test file prepared in step 4' from *the test set* which is a set of question annotated with their coarse and fine classes for every question in the set. We feed the test file to the model and we compare its result by the original annotated classes. The ratio between the correct answers of the model by the number of questions in the test file is the accuracy ratio. If this later is acceptable the model is ready to be used for QA system tasks. The training phase has to be redone if the models accuracy is not sufficient. In that case adjusting SVM parameters is crucial. This process is repeated until the decision model is accurate enough.

3.2. Questions Processing

The first step allows the system to identify the question type of given question. Nevertheless, knowing the question type alone is not sufficient to find the relevant answer of the question neither in DBpedia nor in any other corpora. For example, *what* questions can be quite ambiguous for the classifier in term of the sought answer. So, to deal with that kind of ambiguity,

an additional component which extracts the question focus is necessary. The question focus in our case is represented by **Resources** and **Keywords**. Hence, we need to build two components the first extracts resources and the second extracts Keywords from the question (see Figure 5).

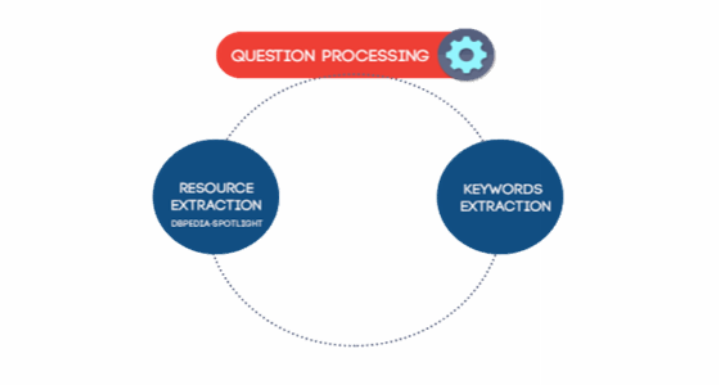

Figure 5: Question processing

As an illustrative example, let's have the question "What is the largest city in China?" The question classifier will return the coarse class "LOC" and the fine class "City". This information is not enough, thus, we need to find the focus of the question which is "largest city" as the keyword and "China" as the resource.

3.2.1. Resource Extraction

Our system use DBpedia spotlight [9] to extract resources from the question. In fact, DBpedia spotlight is a tool designed for automatically annotating mentions of DBpedia resources in text. It provides a solution for linking structured information sources to the Linked Open Data cloud through DBpedia.

Once the set of resources is retrieved from the question, we select only resource compatible with the course and the fine classes. For example if a question has the fine class "country", then we have to focus only on location resources by adjusting DBpedia spotlight (to: only Per. Loc. Org. for example).

3.2.2. Keywords Extraction

In order to extract keywords from the question, the system chooses any word in the question satisfying one of the following seven conditions [10]*:*

1. All non-stop words in quotations.
2. Words recognized as proper nouns by DBpedia Spotlight.
3. Complex nominal and its adjective modifiers.
4. All other complex nominals
5. All nouns and their adjectival modifiers
6. All other nouns
7. All verbs

The schema depicted in (Figure 6) resume the process of "resources and features" extraction. To illustrate this process, we give two examples:

- Example 1: for the question "What is the largest city in China?"
  - **First**, we select all non-stop words in quotations: Largest, City, China.

- **Second**, select words recognized as proper nouns (DBpedia Spotlight): China, we save word "China" as resource.
- **finally**, then we apply chunking on the question and we get:
  ([the_DT largest_JJS city_NN]) in_IN ([China_NNP])?_.

So, we save "the_largest_city" as the keyword after removing the stop-word "the".

- Example2: let's have the question "how tall is Lionel Messi?"

- **First**, select all non-stop words in question: tall, Lionel Messi.
- **Second**, select words recognized as proper nouns by DBpedia Spotlight: Lionel Messi.
- So the Key word is the "tall". We can enrich keywords by extracting all synonyms of the word "tall" and we get:

Altitudinous	lofty	high	towering

- The synonym sets retrieved from merriam-webster (merriam-webster.com) dictionary could be used to expand the set of question keywords.

After retrieving question keywords and resources, the system moves to the step of Query formulation and execution.

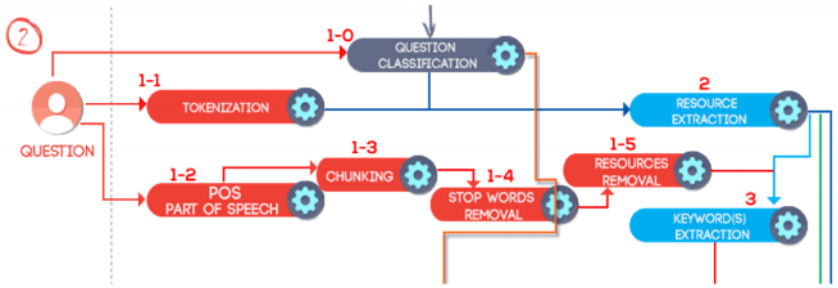

Figure 6: Resource and feature extraction

3.3. Query Formulation and Execution

Answer type, resources and keywords aren't the only needed elements to retrieve the answer from DBpedia. But also determination of ontology classes and properties from keyword set is needed. These later are crucial to build the appropriate SPARQL queries before interrogating the DBpedia server. We schematize these steps in (Figure 7).

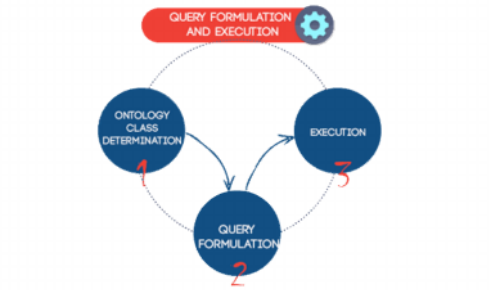

Figure 7: Query Formulation and Execution

3.3.1 Ontology Class Determination

In this step, the system should determine ontology classes and their properties which will be used to construct the final SPARQL query. For illustration, let's take two examples:

Question 1: How tall is Lionel Messi?

Question 2: How many people in Algeria?

The resource of the first question is “Lionel Messi”. And “Algeria” is the resource of the second question.

Now, to retrieve the DBpedia ontology classes and properties we have to build a SPARQL query with the resource itself. For our examples the two queries are the following:

```
SELECT ?Property?h WHERE
{
   <http://dbpedia.org/resource/Lionel_Messi> ?Property?h .
}
```

```
SELECT ?Property?h WHERE
{
   <http://dbpedia.org/resource/Algeria > ?Property?h .
}
```

The result of the query is an RDF file which holds ontology classes and properties and other information belonging to that resource.

The system parses the Query result (RDF file) and retrieves all the ontology classes and properties figuring there. To determine which ontology class to be used in the final SPARQL query, we compute similarity between the keywords and the ontology classes and properties. Thus, similar classes and properties are selected.

3.3.2. Query formulation

Once ontology classes and properties are well fixed, it’s time to build the appropriate SPARQL query to retrieve the answer from DBpedia. The query is composed from resources and ontology classes determined by the keyword set.

To illustrate this step, let’s back to the previous question “how many people in Algeria?” as we seen previously we got:

Recourses*:* Algeria
Keywords: people, colonize, settle, populate, colonize, settle, populate, depopulate, unpeople.
Ontology class*:* populationCensus.
Thus, the system formulates the SPARQL query from ontology classes and properties as follows:

```
select distinct * where {
   <http://dbpedia.org/resource/Algeria >
   <http://dbpedia.org/ontology/populationCensus > ?v.
   OPTIONAL { filter (lang(?v)="en")}
}
```

3.3.3. Execution

Finally, the system interrogates the DBpedia Server[9] to get response to the query built in the previous step. The response of the server is received as an RDF file. This later is parsed to get the answer of the given question. The type of the retrieved answer should belong to the coarse class identified during question classification.

3.4 Illustrative Example

To illustrate the logic implemented
Question: What is the highest place of Karakoram?
Results of tokenization, Part Of Speech POS tagging, chunking and stop words removing are:

Tokens: What, is, the, highest, place, of, Karakoram?
POS: WP, VBZ, DT, JJS, NN, IN, NN,
Chunk: what, is, the_highest_place, of, karakoram.
(B-NP, B-VP, B-NP, I-NP, I-NP, B-PP, B-NP, O)
Stop-words: is, of, the

The result of feature extraction is:

Feature set: What, highest, place, Karakoram, what_highest, highest_place,place_Karakoram,What_highest_place, highest_place_Karakoram, What_ highest_FBG, What_highest_place _FTG, WP, VBZ, DT, JJS, NN, IN, NN, WP_VBZ_FBG, WP_VBZ_DT_FTG, B-NP, B-VP, B-NP, I-NP, I-NP, B-PP, B-NP, O, B-NP_B-VP_FBG, B-NP_B-VP_B-NP _FTG, 6_L,

The result of question classification is:

Coarse class: LOC (Location)
Fine class: other

The results of resources and keywords extraction are:

Focus: Recourses: Karakoram
Keywords:highest_place, altitudinous, lofty, tall, towering, locality, location, locus, point, position,
site, spot, venue, where.

Results of ontology class determination are:

OWL Classes:
ON_highestPlace,PR_highest,PR_highestLatD,PR_highestLatM
PR_highestLatS,PR_highestLatNs,PR_highestLongEw,PR_highestElevation,
PR_highestLongD,PR_highestLongM,PR_highestLongS
ON_highestPosition,PR_p,PR_s,PR_t,PR_hasPhotoCollection
PR_imageCaption,PR_wikiPageUsesTemplate,ON_abstract
ON_maximumElevation,ON_thumbnail,ON_region,PR_region
PR_name,ON_wikiPageExternalLink,PR_map,ON_border,PR_border
PR_mapCaption,ON_country,PR_country,PR_name
Best similar Class: ON_highestPlace **(0.92)**

[9] http://spotlight.dbpedia.org:80/

The final SPARQL query is:

SPARQL Query:

```
select distinct * WHERE
 {
  <http://dbpedia.org/resource/Karakoram>
  <http://dbpedia.org/ontology/highestPlace> ?v .
  OPTIONAL { filter (lang(?v)="en")} .
 }
```

The execution of the query done:

Answer: K2
More information: K2 is just 237 m (778 ft) lower than the 8,848 m (29,029 ft) tall Mount Everest…

3.4 Global schema

By encompassing the three steps detailed previously, we obtain the global architecture of our proposed 'DBpedia based factoid question answering System'. The system starts by receiving question from the system user, and finishing by providing the appropriate answer for that question, passing through the different phases explained previously and depicted in (figure 8)

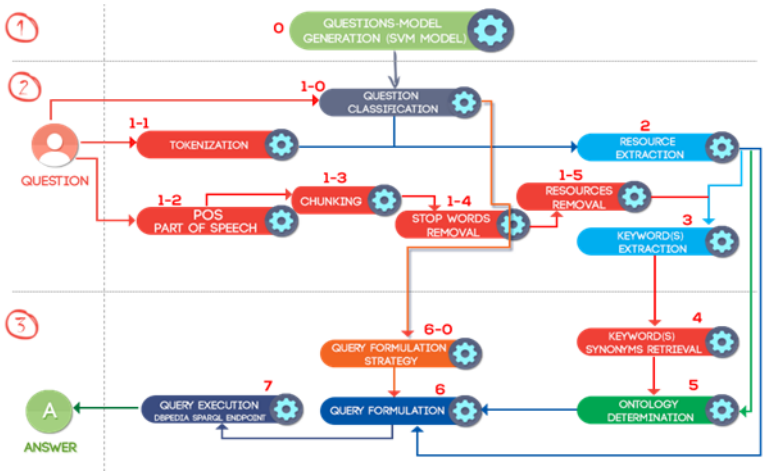

Figure 8: Global schema of system Architecture

4. SELNI System Implementation

SELNI system is a java-based implementation of the architecture described previously. It's a sentence-level question answering system that returns answers for factoid questions based on the DBpedia ontology. The system is coined SELNI which means ask me in the Arabic language. A set of tools, Languages and APIs were used in the development of the SELNI system. The table (table 3) lists these elements.

JAVA Programing Language[10]	Eclipse[11] IDE 4.2.2
Stanford Part-Of-Speech Tagger[12] Java API	SPARQL[13] (RDF query) Language

[10] http://www.oracle.com/us/java/index.html
[11] http://www.eclipse.org/
[12] http://nlp.stanford.edu/software/tagger.shtml

JENA[14] Library Semantic Web building framework	OpenNLP[15] a Natural Processing Language API
LIBSVM[16] (LibLinear): SVM Java library	Snowball[17] : JAVA API for stemming
Merriem-Webster[18] : online dictionary	Jsoup[19] : Java API for Html parsing
OpenCCG[20]: a GCC parser	SWT[21]: GUI java API
DBpedia Spotlight[22] : a tool for automatically annotating mentions DBpedia resources in texts	

Table 3: Tools, languages and APIs used in Selni system developments

Figure 9: screenshot of SELNI.

5. TUNING, EXPERIMENTS END EVALUATION

During the development of the SELNI system, it was crucial to get a very good accuracy ratio by training its system classifier. As we presented in the previous sections, we used the SVM algorithm to train our proposed system. The SVM algorithm holds many parameters that need tuning and adjustments.

[13] http://www.w3.org/TR/rdf-sparql-query/
[14] http://jena.apache.org/about_jena/about.html
[15] http://opennlp.apache.org/
[16] http://liblinear.bwaldvogel.de/
[17] http://snowball.tartarus.org/index.php
[18] http://www.merriam-webster.com/
[19] http://jsoup.org/
[20] http://groups.inf.ed.ac.uk/ccg/software.html
[21] http://www.eclipse.org/swt/
[22] https://github.com/dbpedia-spotlight/dbpedia-spotlight/wiki

First, Tuning SVM parameter strongly depends on the chosen SVM kernel. LibSVM provides four kernels functions to be used. These kernels are listed in table (table 4).

Kernel	Function
Linear	u'*v
Polynomial	(gamma*u'*v + coef0)^degree
Radial basis function	exp(-gamma*|u-v|^2)
Sigmoid	tanh(gamma*u'*v + coef0)

Table 4: Type of kernel function tested

Hence, we run many training with different kernels. Finally, we decided to use the linear Kernel implemented by the Library "LibLinear". This kernel is faster than other kernels against the very large dataset that we have. Using this kernel and after many adjustments of its associated parameters, we got an accuracy ration of 87% which is acceptable in question classification.

5.1 SELNI evaluation

In order to evaluate the SELNI system we used a set of 200 factoid question collected from different sources. The results obtained by our QA are compared to results obtained by the QAKis[17], and True_knowledge (evi[23]) [18] for the same question dataset. We selected this two question answering systems because QAkis uses the same knowledge base (DBPedia) as Selni. And, True_knowledge QA uses its own knowledge base which holds 24 283,511,156 facts about 9,237,091 things. Unfortunately, we could not find the implementation of the QASYO [12] system, which uses the Yago ontology, to compare its results with our QA system.

The evaluation has been done manually (see table 5 for question samples). We found that SELNI and QAKis systems could answer for 72 questions and both could not answer for 36 questions. Selni system answered 136 questions and failed in answering for 64 questions. Also, SELNI answered for 64 questions that QAKis could not answer, and 48 questions that True knowledge could not answer.

From it side, QAKis answered for 100 questions and failed in answering for the remaining 100 questions. It answers only for 28 questions that our system could not answer and 36 questions that True knowledge could not answer. Both QAKis and True knowledge answered only 12 questions that SELNI could not answer.

The True knowledge answered for 92 questions and failed to answer 108 questions. It answers only for 12 questions that SELNI could not answer and 28 questions that QAKis can't answers.

N	Question	SELNI	QAKis	TrueKnowledge
1	Why is the sky blue?	+	-	+
2	What is TCP/IP?	+	-	+
3	Which organizations were founded in 1950?	-	+	+
4	Who directed the film "Fail Safe"?	+	-	+
5	Who is the director of the mentalist?	+	+	-

[23] Trueknowledge became Evi : www.evi.com
[24] Statistics of august 2010

6	Who founded Google?	+	+	+
7	When was universal studios founded?	+	+	+
8	Give me all actors starring in Batman Begins.	-	+	-
9	How far is Neptune from the sun?	-	-	+
10	Who was the first person on the Moon?	-	-	-
11	Give me all female Russian astronauts.	-	-	-
12	what is the name of barrack Obama's wife	+	-	-
13	Who engineered the leaning tower of Pisa?	+	-	-
14	How many students in university of Manitoba?	+	-	-

(+) refers to answer & (-) refers to "does not answer"
Table 5: Sample questions

The evaluation of selni results are presented in table 7.

answered	Right Answered	Precision	Recall	Accuracy	F-Measure
200/200	72/200	0.66	0.5	0.5	0.568

Table 7: SELNI performances on the selected Dataset

Most of mistakes made by SELNI are due to one of the following reasons:

- Absence of the information in DBpedia ontology
- The formulation of the query with resources that should be represented as keyword. Or vice-versa and that's caused by the DBpedia spotlight annotations.
- Question classification error.

The results shows that SELNI gives for this question set better results than QAKis and true knowledge systems do. This evaluation shows that the proposed system gives encouraging results. And with some enhancement and tweaks on the system, we can rich better accuracy result

4. CONCLUSION AND FUTURE WORK

The architecture and the implementation of a new question answering system were discussed in this paper. The proposed system uses the SVM algorithm to train its question classifier. Questions of the TREC10 dataset were used to train the classifier. This later reaches an accuracy of 86% based on TREC 10 test set. In addition, the proposed system interrogates the DBpedia server by SPARQL queries to retrieve the correct answer that belongs to the answer type class identified during question classification. A java-based implementation of the system is coined by 'SELNI'.
Moreover, the developed System was evaluated with a set of factoid questions. We compared results obtained by our system against results obtained from another question answering systems. We found that SELNI system offers encouraging results. Nevertheless, SELNI cannot answer all factoid questions due to the absence of the information in DBpedia knowledge framework, or due to the quality of the decision model used by the system. Consequently, enhancements have to be done on the system in order to achieve better results. It's crucial to use other knowledge bases and ontologies like YAGO. In addition, it's possible to use IR techniques with search engine results to improve answers retrieval.

In future extensions, the system could be customized in different ways; as a desktop, Web service, mobile application, web application or even as a java API.

Finally, we look for building an extension to SELNI question answering system that uses other languages for questioning and answering.

REFERENCES

[1] Salton, Gerard & McGill, Michael J. (1986) *Introduction to modern Information Retrieval*, McCraw-Hill Inc., USA.

[2] Hirschman, Lynette & Gaizauskas, Rob. (2001) *Natural language question answering: the view from here*, Natural Language Engineering 7 (4): 275-300., Cambridge University press.

[3] Azari, David & al (2012) *Web-Based Question Answering: A Decision-Making Perspective*. Proceedings of the Nineteenth Conference on Uncertainty in Artificial Intelligence

[4] Wenyin, Liu & al (2009) *A Web-Based Platform for User-Interactive Question Answering.* World Wide Web12.

[5] Ren, Han & al (2011) *A web knowledge based approach for complex question answering*. In *Proceedings of the 7th Asia conference on Information Retrieval Technology* (AIRS'11).

[6] Lu, Wenpeng & al (2012) *Question Answering System Based on Web*, In Proceedings of the 2012 Fifth International Conference on Intelligent Computation Technology and Automation (ICICTA '12).

[8] Perera, Rivindu (2012) *IPedagogy: Question Answering System Based on Web Information Clustering*, In Proceedings of the 2012 IEEE Fourth International Conference on Technology for Education (T4E '12). IEEE Computer Society, Washington, DC, USA

[9] Tartir, Samir. McKnight, Bobby & Arpinar, I. Budak (2009) *SemanticQA: web-based ontology-driven question answering*, In Proceedings of the 2009 ACM symposium on Applied Computing (SAC '09).

[10] Guo, Qinglin & Zhang, Ming (2008) *Question answering system based on ontology and semantic web*, In Proceedings of the 3rd international conference on Rough sets and knowledge technology(RSKT'08)

[11] Adolphs, Peter. Theobald, Martin. Schafer, Ulrich. Uszkoreit, Hans & Weikum, Gerhard (2011) *YAGO-QA: Answering Questions by Structured Knowledge Queries*, In Proceedings of the 2011 IEEE Fifth International Conference on Semantic Computing (ICSC '11). IEEE Computer Society, Washington, DC, USA

[12] Moussa, Abdullah M. & Rehab, Abdel-Kader (2011) *QASYO: A Question Answering System for YAGO Ontology*. International Journal of Database Theory and Application. Vol. 4, No. 2, June, 2011. 99.

[13] Auer, S. Bizer, C. Kobilarov, G. Lehmann, J. & Ives, Z (2007) *DBpedia: A Nucleus for a Web of Open Data*, in In 6th International Semantic Web Conference, Busan, Korea. Springer, 2007, pp. 11–15.

[14] Morsey, M. Lehmann, J. Auer, Sö. Stadler, C. & Hellmann, S. (2012) *DBpedia and the Live Extraction of Structured Data from Wikipedia*. Program: electronic library and information systems, 46, 27.

[15] Li, Xin & Roth, Dan. (2002) *Learning question classifiers*. In Proceedings of the 19th international conference on Computational linguistics - Volume 1 (COLING '02), Vol.1 Association for Computational Linguistics, Stroudsburg, PA, USA

[16] Glenn M. Fung and O. L. Mangasarian (2005) *Multicategory Proximal Support Vector Machine Classifiers*. Maching Learning. 59, 1-2.

[17] Cabrio, Elena. Cojan, Julien. Aprosio, Alessio Palmero. Magnini, Bernardo. Lavelli, Alberto & Gandon, Fabien (2012) QAKiS: an Open Domain QA System based on Relational Patterns Proceedings of the ISWC 2012 Posters & Demonstrations Track, Boston, USA

[18] Tunstall-Pedoe, William (2010) *True Knowledge: Open-Domain Question Answering Using Structured Knowledge and Inference*, AI Magazine, Vol 31 Number 3 P 80-92