AlQuAnS – An Arabic Language Question Answering System

Mohamed Nabil, Ahmed Abdelmegied, Yasmin Ayman, Ahmed Fathy, Ghada Khairy,
Mohammed Yousri, Nagwa El-Makky and Khaled Nagi
Computer and Systems Engineering Department, Faculty of Engineering, Alexandria University, Alexandria, Egypt

Keywords: Arabic Question Answering Systems, Arabic Morphological Analysis, Question Analysis, Question Classification, Answer Extraction.

Abstract: Building Arabic Question Answering systems is a challenging problem compared to their English counterparts due to several limitations inherent in the Arabic language and the scarceness of available Arabic training datasets. In our proposed Arabic Question Answering system, we combine several previously successful algorithms and add a novel approach to the answer extraction process that has not been used by any Arabic Question Answering system before. We use the state-of-the-art MADAMIRA Arabic morphological analyser for preprocessing questions and retrieved passages. We also enhance and extend the question classification and use the Explicit Semantic Approach (ESA) in the passage retrieval process to rank passages that most probably contain the correct answer. We also introduce a new answer extraction pattern, which matches the patterns formed according to the question type with the sentences in the retrieved passages in order to provide the correct answer. A performance evaluation study shows that our system gives promising results compared to other existing Arabic Question Answering systems, especially with the newly introduced answer extraction module.

1 INTRODUCTION

Traditionally, search engines provide users with documents relevant to their search requests. The request is usually in the form of a list of keywords. However, the users must take the trouble of searching for the exact answer to their question inside each of the retrieved documents. Nowadays, a new approach matches the user needs by analyzing the question posted in the search field from a linguistic point of view, attempting to understand what the user really means and extracting the correct answer to the user question from the retrieved documents.

Recently, Question Answering (QA) has been one of the main focal points of research in the area of natural language processing. Some great efforts were made to provide reliable QA systems for different languages. Unfortunately, Arabic QA systems are still not in the mainstream even though Arabic is one of the six official languages of the United Nations and it is the main language of most of the Middle East countries. In fact, the Arabic language ranks *fifth* in the world's league table of languages, with an estimated 300 million native speakers.

Very few attempts were made to investigate Arabic QA because Arabic is a complex language and is very hard to be analyzed by language processors. Arabic enjoys a very complex morphology that needs very intelligent morphological analysis subsystems. By morphological analysis, it is meant the process of assigning the morphological features of a word; such as:

- its root or stem,
- its morphological pattern,
- its part-of-speech (noun, verb or particle)
- its number (singular, dual or plural)
- its case or mood (nominative, accusative, genitive or jussive)

to the word. The root-patterned nonlinear morphology of Arabic makes both theoretical and computational processing for Arabic text extremely hard.

To begin with, Arabic has a completely different orthography based on standard Arabic script going from right to left. Each letter has *three* different shapes depending on its position within the word. Each letter has a *diacritic* sign above or below the letter. Changing the diacritic of one letter may change the meaning of the whole word. Printed and online

Nabil M., Abdelmegied A., Ayman Y., Fathy A., Khairy G., Yousri M., El-Makky N. and Nagi K.
AlQuAnS âĂŞ An Arabic Language Question Answering System.
DOI: 10.5220/0006602901440154
In *Proceedings of the 9th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (KDIR 2017)*, pages 144-154
ISBN: 978-989-758-271-4

text come usually *without* diacritics leaving plenty of room for word ambiguity.

Arabic is also a highly *derivational* language. It is a highly *inflectional* language as well.

`Word = prefix(es) + lemma + suffix(es).`

The prefixes can be articles, prepositions or conjunctions; whereas the suffixes are generally objects or personal/possessive anaphora. Both prefixes and suffixes can be combined, and thus a word can have zero or more affixes. Figure 1 shows an example of the composition of an Arabic word.

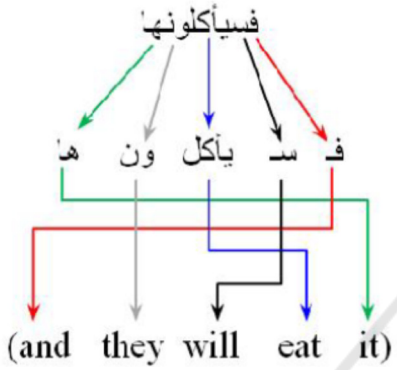

Figure 1: Example Arabic inflection (Benajiba and Rosso, 2007b).

The absence of capital letters is another challenge in the Named Entity Recognition (NER) in Arabic (Benajiba and Rosso, 2007a) and (Benajiba et al., 2007). Lots of Arabic names are adjectives; such as, “gameel” (handsame), “zaki” (intelligent), or “khaled” (immortal).

Last but not least, from a statistical viewpoint, if Arabic texts are compared to texts written in other languages which have a less complex morphology, Arabic texts look much more sparse because of the inflectional characteristic of the language that we mentioned above. It is this specific characteristic of the language which makes it more difficult to tackle in each of the Natural Language Processing (NLP) tasks.

The above challenges are enough to motivate us to develop AlQuAnS, a new system that can extract an accurate answer to the user’s questions. We present a complete Arabic language Question Answering System, that understands natural Arabic questions and extracts their answers from the retrieved documents found on the WWW. The system gives the user a short precise answer to their question, instead of just giving them hundreds of documents to search in manually. The main design goals can be summarized in the following points:

- *Pre-processing* the question to make the data retrieval more accurate.
- Building a question *classification* module using a suitable classification technique to classify the input question into a certain type, and hence produce the expected answer type.
- Building a *semantic information retrieval* module capable of retrieving the related documents.
- *Extracting* the ranked answers to the input questions from the retrieved documents with high accuracy.

The objective of our work is to contribute in the improvement of Arabic QA systems by enhancing the passage retrieval process module. We propose two directions for such enhancement: firstly, a semantic query expansion is used to achieve a high level of completeness (recall) when the information retrieval process retrieves passages; then a semantic-based process (ESA) is used for passage re-ranking in order to have the expected answer at the top of the candidate passages list. Finally, an answer extraction module extracts the answer from the top candidate passages.

The system has to perform well by providing *correct* answers. So, we benchmark AlQuAnS against similar Arabic QA systems found in literature. For that, we managed to get their same training and benchmarking datasets. We update these datasets to adapt the answers since search engines deliver different; yet correct; results over time. For example, "*How many Syrian refugees live in Jordan?*". The answer changes each year. Other results are more or less strict. For example, "*Where was Ibn Batota born?*" The expected answer in the datasets used by the researchers is "*Tanjier*". However, another accepted answer can be "*Morocco*". For a fair comparison, we compare the quality of individual system components and not only the overall quality.

The rest of the paper is organized as follows. In Section 2, we give a short survey on the related standard work in the field of Arabic QA systems. Section 3 contains an overview of the system architecture of AlQuAnS and detailed description of each system component. The system evaluation is presented in Section 4. Section 5 concludes the paper and presents some ideas for our future work in this area.

2 RELATED WORK

A Question Answering system is a system that takes an input question from the user, retrieves the related result sets to the question topic and then extracts an

exact answer to the question to be returned to the user. A typical state-of-the-art information-retrieval-based Question Answering system, divides the Question Answering task into *three* core components:

- Question Analysis (including Question preprocessing and classification),
- Information Retrieval (or passage retrieval),
- Answer Extraction.

Question classification plays an essential role in QA systems by classifying the submitted question according to its type. Information retrieval is very important for question answering, because if no correct answers are present in a document, no further processing can be carried out to find the answer. Finally, answer extraction aims at retrieving the correct passage containing the answer within the retrieved document.

2.1 Arabic QA Systems

QARAB (Hammo, et al. 2002) is a QA system to support the Arabic language. The system is based on the *three*-module generic architecture:

- question analysis,
- passage retrieval, and
- answer extraction.

It extracts the answer from a collection of Arabic newspaper text. For that, it uses a keyword matching strategy along with matching simple structures extracted from both the question and the candidate documents selected by the information retrieval module using an existing tagger to identify proper names and other crucial lexical items. The system builds lexical entries for them on the fly. For system validation, four native Arabic speakers with university education presented 113 questions to the system and judged whether the answers of the system are correct or not.

The Arabic language was introduced for the first time in 2012 in the QA4MRE lab at CLEF (Trigui et al., 2012). The intension of the research is to ask questions which require a deep knowledge of individual short texts and in which systems are required to choose one answer from multiple answer choices. The work uses shallow information retrieval methods. Unfortunately, the overall accuracy of the system is 0.19 and the questions proposed by CLEF are suitable only for modern Arabic language.

ALQASIM (Ezzeldin et al., 2013) is an Arabic QA selection and validation system that answers multiple choice questions of QA4MRE @ CLEF 2013 test-set. It can be used as a part of the answer validation module of any ordinary Arabic QA system. It comes up with a new approach like the one used by human beings in reading tests. A person would normally read and understand a document thoroughly, and then begins to tackle the questions. So, the suggested approach divides the QA4MRE process into *three* phases:

- document analysis,
- locating questions and answers,
- answer selection.

ArabiQA (Benajiba and Rosso, 2007b) is a QA system that is fully oriented to the modern Arabic language. ArabiQA is obeying to the general norms reported at the CLEF conference. However, the system is not complete yet. The following points is a part of the researchers' investigation, as listed in their work:

- The adaptation of the JIRS passage retrieval system to retrieve passages from Arabic text.
- The development of the annotated ANERcorp to train the Named Entity Recognition (NER) system.
- The development of the ANERsys Named Entity Recognition system for modern Arabic text based on the maximum entropy approach.
- The development of an Answer Extraction module for Arabic text for *factoid* questions (Who, where and when questions).

DefArabicQA (Trigui et al., 2010) presents a definitional QA system for the Arabic language. The system outperforms the use of web searching by two criteria. It permits the user to ask an ordinary question (e.g.,"*What is X?*") instead of typing in a keyword-based query. It then attempts to return an accurate answer instead of mining the web results for the expected information. The question topic is identified by using *two* lexical question patterns and the answer type expected is deduced from the *interrogative pronoun* of the question. Definition ranking is performed according to three scores: a pattern weight criterion, a snippet position criterion, and a word frequency criterion.

The IDRAAQ (Abouenour, 2012) system is another Arabic QA system based on *query expansion* and *passage retrieval*. It aims at enhancing the quality of retrieved passages with respect to a given question. In this system, a question analysis and classification module to extract the keywords, identify the structure of the expected answer and form the query to be passed to the Passage Retrieval (PR) module. The PR

extracts a list of passages from an Information Retrieval process. Thereafter, this module performs a ranking process to improve the relevance of the candidate passages. Finally, the Answer Validation (AV) module validates an answer from a list of candidate answers.

Al-Bayan (Abdelnasser, 2014) is a *domain specific* Arabic QA system for the Holy Quran. It takes an Arabic question as input and retrieves semantically relevant verses as candidate passages. Then, an answer extraction module extracts the answer from verses obtained accompanied by their Tafseer (standard explanations of Quran). The system has *four* functionalities:

- It merges *two* Quranic ontologies and uses *two* Tafseer books.
- It applies a semantic search technique for information retrieval.
- It applies a state-of-the-art technique (SVM) for question classification.
- It builds Quranic-based training data sets for classification and Named Entities Recognition (NER).

2.2 The Need to Extend Related Work

From the above presentation of related work in Arabic QA systems, it appears to us that no single research provides a solution that is applicable for *open domain*, provides a good *query expansion* functionality and *extracts answers* from document snippets with relatively high accuracy.

3 SYSTEM ARCHITECTURE

3.1 Overview

Our proposed system is an Arabic language Question Answering System that requires no specific for domain knowledge. Figure 2 shows our system architecture consisting of an *offline* and an *online* part.

The *semantic interpreter* (Gabrilovich and Markovitch, 2007) is built during the offline phase, where 11,000 Arabic Wikipedia documents are first pre-processed by MADAMIRA (Pasha et al., 2014) and then a weighted inverted index is built for them using Lucene (McCandless et al., 2010). The built semantic interpreter is used by the *Passage Retrieval* (PR) module in the online phase to rank retrieved passages.

In the online part, the input question passes through a *pre-processing* module, which is built on MADAMIRA (Pasha et al., 2014) as well. Then, the processed question is fed to the *Question Analysis* module; which is composed of a *Query Expansion* (QE) part and a *Question Classification* (QC) part. The QE submodule is responsible for expanding the query to include its other morphological forms. The QC submodule is responsible for classifying questions into types, e.g., who, when, where. In the *Information Retrieval* module, the semantically relevant documents are retrieved using the *Online Search Engine* and ranked by the Explicit Semantic Analysis (ESA) approach. Finally, an *Answer Extraction* (AE) module extracts the correct answer from the documents obtained according to the patterns provided by the *pattern construction* module; which builds its patterns from the training dataset using a set of features shown in section 3.5.

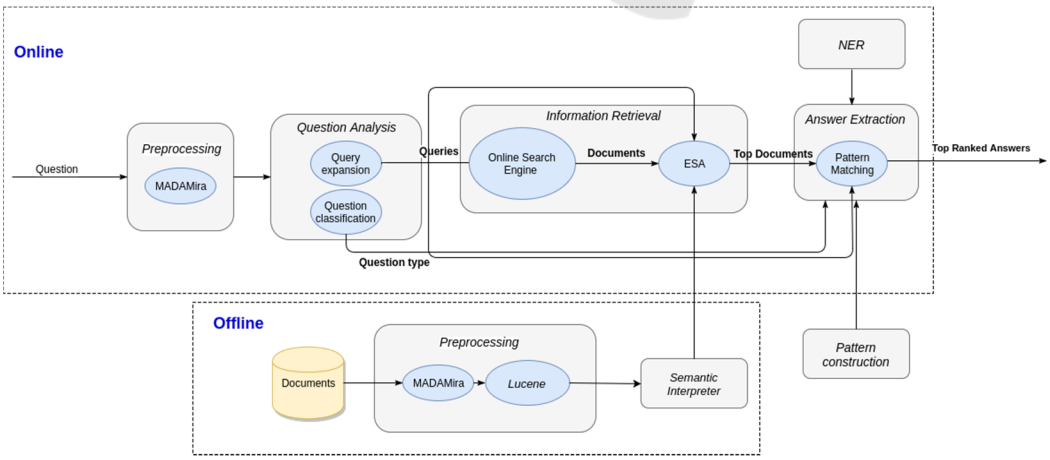

Figure 2: Overall system architecture.

3.2 Pre-Processing Operations

Text Preprocessing is done by applying morphological analysis software to identify the structure of the text. Typical operations include:

- normalization,
- stemming,
- Part-Of-Speech (POS) tagging, and
- stop words removal.

Morphologically, the Arabic language is one of the most complex and rich languages. Thus, morphological analysis of the Arabic language is one of the complex tasks that has been popular in recent research. In this module, we rely heavily on MADAMIRA (Pasha et al., 2014). MADAMIRA combines the best aspects of two previously commonly used systems for Arabic pre-processing, MADA found in (Habash et al., 2009) and AMIRA found (Diab, 2009).

MADA is a system for Morphological Analysis and Disambiguation for Arabic. The primary purpose of MADA is to, given raw Arabic text, derive as much linguistic information as possible about each word in the text, thereby reducing or eliminating any ambiguity surrounding the word. MADA also includes TOKAN, a general tokenizer for MADA-disambiguated text (Habash et al., 2009). TOKAN uses the information generated by the MADA component to tokenize each word according to a highly- customizable scheme.

AMIRA is a system for tokenization, part-of-speech tagging, Base Phrase Chunking (BPC) and Named Entity Recognition (NER).

3.3 Question Analysis

The Question Analysis module consists of two submodules: *Query Expansion* (QE) and *Question Classification* (QC).

3.3.1 Query Expansion

This submodule is based on the Query Expansion module of (Abouenour et al., 2010). In this submodule, the content and the semantic relations of the Arabic WordNet (AWN) ontology (Elkateb et al., 2016) are used. The AWN ontology is a free resource for modern standard Arabic. It is based on the design and the content of Princeton WordNet (PWN) (Fellbaum, 2005). It has a structure similar to WordNets existing for approximately 40 languages. It is also connected to the Super Upper Merged Ontology (SUMO) (Niles and Pease, 2003). SUMO is an upper level ontology which provides definitions for general purpose terms and acts as a foundation for more specific domain ontologies. It contains about 2,000 concepts. The AWN data is divided into *four* entities, as shown in Figure 3.

- *Items* are conceptual entities, including synsets, ontology classes and instances.
- *Word* entity is a word sense, where the citation form of the word is associated with an item.
- A *Form* is a special form that is considered dictionary information. The forms of Arabic words that go in this entity are the root and/or the broken plural form, where applicable.
- A *link* relates two items, and has a type such as equivalence, subsuming, etc. Links connect sense items to other sense items, e.g. a PWN synset to an AWN synset; a synset to a SUMO concept, etc.

The current release of AWN contains 11,270 Arabic synsets (versus 115,000 synsets for English WordNet), 23,496 Arabic words (versus 200,000 words for English WordNet). It contains also entries that are named entities (1,142 synsets and 1,648 words) (Abouenour et al., 2010). The AWN ontology contains different relations between its items such as hyperonymy/hyponymy (supertypes or subtypes relations), synonymy, meronymy/holonymy (part/whole relations).

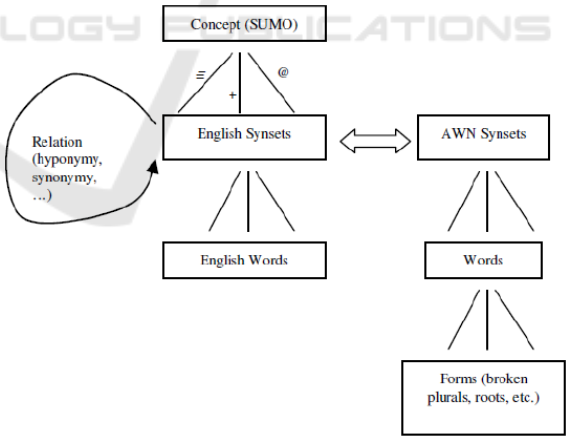

Figure 3: The AWN data structure.

Our semantic QE approach uses *four* semantic relations among those existing between AWN synsets (items), words and forms. Therefore, the approach defines four sub-processes for the query expansion:

- QE by synonyms,
- QE by definitions,
- QE by subtypes, and
- QE by supertypes.

Unlike QE by subtypes and supertypes, the QE by synonyms and definitions apply our semantic expansion in a way to have new terms related to the considered keyword. After that, we re-rank the passages to have in the first ranks those containing both the question keywords and the new generated terms close each to another. Our QE process is applied only for keywords which are not stopwords, namely: ما (what), هو (he) and الذي (that).

3.3.2 Question Classification

In this stage, we classify the question to the anticipated type of the answer. For example, the question من أسس جوجل؟ (who founded Google?) should be classified into the type *human individual*. This information would narrow down the search space to identify the correct answer. Additionally, this information implies different strategies to search and verify the candidate answer. The derivation of expected answer types is often carried out by means of machine learning approaches. This task relies on *three* parts:

- taxonomy of answer types into which questions are to be classified,
- a corpus of questions prepared with the correct answer type classification, and
- an algorithm that learns to make the actual predictions given this corpus.

In our system, we use the Support Vector Machines (SVM) classifier since it has shown to produce the best results during our experiments. Question classification needs a taxonomy to classify question types. We base our classification on the work of Li and Roth (Li and Roth, 2002). Their work provides a hierarchical classifier, taxonomy and data to be used in English question classification. Since 2002, their work has been used by all researchers who are interested in building QA systems. They propose a two-layered question taxonomy which contains six coarse grained categories:

- Abbreviation
- Description
- Entity
- Human
- Location
- Numeric value

and 50 fine grained categories.

We use the same coarse taxonomy (ABB, EXP, HUM, ENT, LOC, NUM) and build a classifier model to test input questions against that model. However, due to the limitation in the Answer Extraction module (AE), we have to limit the taxonomy to LocationCity, LocationCountry, HumanIndividual, NumericDate). With these *four* sub-categories, we focus more on a QA system that can answer questions that ask for cities, countries, humans individuals and different kinds of dates (birthdays, event dates, etc.). The system yields good results in this classification.

For testing and training, we use a data set that consists of 230 classified questions divided into 180 questions used for training and 50 questions used for testing. The data does not contain all types of questions. The set includes the Arabic questions: *where*, *who*, *what* (followed by verb), *how much*, and *where* but does not include: *what* (followed by pronoun), *why* and *how*. However, adding these types of questions to the classifier only requires adding them to the training dataset.

3.4 Information Retrieval

The Information Retrieval module consists of two sub-modules: the *Online Search Engine* and the *Passage Retrieval* submodules.

Our system is designed to interface with commonly available search engine modules. However, in our implementation, we choose the Yahoo API to be numerically comparable to previous systems using the same API, e.g., (Abouenour et al., 2010).

For the *Passage Retrieval* submodule to function, we construct a general *Semantic Interpreter* (SI) that can represent text meaning. we use Wikipedia as a *repository* of basic concepts. Logically, we choose Wikipedia for several reasons including the following.

- It includes concepts in a large variety of topics.
- It is constantly maintained and extended by a huge community.
- Since the goal is to interpret natural language, we like the concepts to be natural, i.e, can be recognized and used by human beings.
    - Each concept C_i has an associated document d_i, so that we can easily determine the strength of its affinity with each term in the language.

3.4.1 Building the Semantic Interpreter

We use the Explicit Semantic Analysis (ESA) approach proposed in (Gabrilovich and Markovitch, 2007). Given a set of concepts, C_1, ..., C_n, and a set of associated documents, d_1, ..., d_n, we build a sparse table T where each of the n columns corresponds to a concept, and each of the rows corresponds to a word that occurs in $\cup_{i=1...n} d_i$. An entry $T[i, j]$ in the table

corresponds to the term frequency–inverse document frequency (tf-idf) value of term t_i in document d_j

$$T\,[i,j] = tf\,(t_i, d_j).\log\frac{n}{d\,f_i} \quad (1)$$

where term frequency is defined as:

$$tf\,(t_i, d_j) = \begin{cases} 1 + \log count(t_i, d_j) & if\ count(t_i, d_j) > 0 \\ 0 & otherwise \end{cases} \quad (2)$$

and $d\,f_i = |\{d_k: t_i \in d_k\}|$ is the number of documents in the collection that contains the term t_i (*document frequency*). Finally, cosine normalization is applied to each row to discard differences in document length:

$$T\,[i,j] \leftarrow \frac{T\,[i,j]}{\sqrt{\sum_{l=1}^{r} T\,[i,j]}} \quad (3)$$

where r is the number of terms.

The semantic interpretation of a word t_i is obtained as a row i of table T. In other words, the meaning of a word is given by a vector of concepts paired with their tf-idf scores, which reflects the relevance of each concept with respect to the word. The semantic interpretation of a text fragment, $\langle t_1, ..., t_k \rangle$, is the centroid of the vectors representing the individual words. This definition allows us to partially perform word sense disambiguation. Consider, for example, the interpretation vector for the term "mouse". It has two sets of strong components, which correspond to two possible meanings: "mouse (rodent)" and "mouse (computing)". Similarly, the interpretation vector of the word "screen" has strong components associated with "window screen" and "computer screen". In a text fragment such as "I purchased a mouse and a screen", summing the two interpretation vectors will boost the computer-related components, effectively disambiguating both words. Table T can also be viewed as an inverted index, which maps each word to a list of concepts where it appears. Inverted index provides a very efficient computation of distance between interpretation vectors.

3.4.2 Using Semantic Interpreter

Explicit Semantic Analysis represents text as interpretation vectors in the high-dimensional space of concepts. With this representation, computing semantic relatedness of texts simply amounts to compare their vectors. Vectors could be compared using a variety of metrics; we use the cosine similarity metric for computing semantic relatedness throughout our performance evaluation.

3.4.3 Computing the Semantic Relatedness

In order to determine the semantic relatedness between the question and the retrieved snippets, we compute the question vector and the vectors of snippets using the *Semantic Interpreter* (SI) obtained. Then, we compute the cosine similarity between the question concept vector and the concept vector of each snippet *i*. The result scores are used to select the top-scoring snippets that are relevant to the question. The more similar the snippet vector to the query vector is, the more likely it is to be related to the query as illustrated in Figure 4.

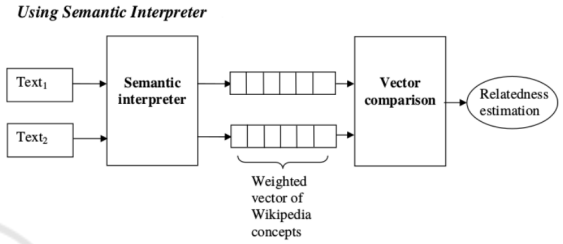

Figure 4: The semantic relatedness between the query and the document (Gabrilovich and Markovitch, 2009).

3.5 Answer Extraction

The purpose of the Answer Extraction (AE) module is to search for candidate answers within the relevant passages and extract the most likely answers. Using certain patterns for each type of question is the main approach. In general, patterns can be written by people or learnt from a training dataset. The type of the expected answers is always taken into consideration. That's why, we use a Named Entity Recognition (NER) system with the patterns extracted for each question type to adapt the approach proposed in (Ravichandran and Hovy, 2002).

Our Answer Extraction module is composed of *three* phases. The first and second phases are based on the approach in (Ravichandran and Hovy, 2002). The *first* phase is to use the web documents retrieved by the Passage Retrieval module to construct a table of patterns for each question type. The *second* phase is to rank these patterns by calculating their corresponding precision. The *third* phase is to find the answer using the extracted answer patterns then filter the answers using the MADAMIRA NER.

3.5.1 Constructing a Table of Patterns

In this phase, we select an example for a given question type having question and answer terms. Then, we submit the question and the answer terms as queries

to the search engine. After downloading the top *m* web documents, we only retain those sentences that contain both the question and the answer terms. Then, we tokenize the input text, smooth variations in white space characters, and remove `html` and other tags and pass the sentences through the suffix tree to get the longest matching substrings containing answer and question term. Finally, we replace question and answer terms with tags and store them as patterns.

3.5.2 Calculating the Precision for Each Pattern

In this phase, we query the search engine by using only the question term. We download the top *m* web documents and segment their documents into individual sentences. We retain only those sentences that contain the question term. For each pattern obtained from the previous phase, we check the presence of the pattern in the sentence and calculate the F-measure of each pattern where,

$$F = 2 \frac{Precision \text{ x } Recall}{(Precision + Recall)} \quad (4)$$

Where *Precision* of each pattern = C_a/C_o and *Recall* of each pattern = $C_a/(C_a + C_n)$; in which:

- C_a = total number of patterns with the answer term present.
- C_o = total number of patterns present with answer term replaced by any word.
- C_n = total number of patterns present with answer term and not having question term.

Finally, we sort the patterns according to their *F*-measures.

3.5.3 Finding Answers

In this last stage, we determine the question type of the new question using the Question Analysis module. We create a query from the question term and pass it to Information Retrieval module. We segment the documents obtained into sentences and smooth out white space variations and `html` and other tags, as we did before. Then, we replace the question term in each sentence by the question tag. Using the pattern table developed for that particular question type, we search for the presence of each pattern. We select words matching the tag <answer> and sort these answers by their pattern precision scores. Finally, we use the MADAMIRA system to recognize the words which would contain the answer in the pattern having the same type of the wanted answer.

3.5.4 Using Suffix Trees

The purpose of the past three phases is to extract the matching answer passage. For that purpose, we use *suffix trees* for extracting substrings of optimal length that contain the question and the answer terms as substrings in it. Suffix Tree is very useful in numerous string processing and computational biology problems. A suffix tree *T* for an *m*-character string *S* is a rooted directed tree with exactly *m* leaves numbered 1 to *m*. A suffix tree has the following characteristics.

- The root can have zero, one or more children.
- Each internal node has at least two children.
- Each edge is labeled with a nonempty substring of *S*.
- No two edges coming out of the node can have edge-labels beginning with the same character.

Concatenation of the edge-labels on the path from the root to leaf *i* gives the suffix of *S* that starts at position *i*, i.e., *S* [*i*...*m*]. For example, Figure 5 is the suffix tree for the string xabxac. The last character has to be unique so the tree is explicit.

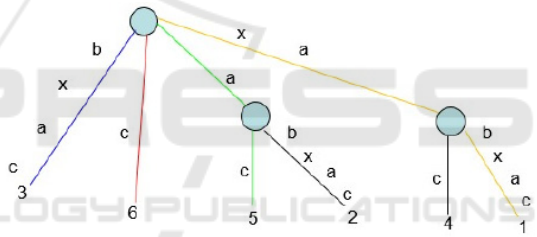

Figure 5: Suffix tree of the string xabxac.

In our implementation, we employ an algorithm which is adapted from the famous Ukkonen's algorithm (Ukkonen, E., 1995). Taking an abstract view, the algorithm constructs an implicit suffix tree T_i for each prefix *S* [1..*i*] of *S* of length *m*. It first builds T_1 using 1st character, then T_2 using 2nd character, then T_3 using 3rd character, . . . , T_m using mth character. Implicit suffix tree T_{i+1} is built on top of implicit suffix tree T_i. Each phase *i*+1 is further divided into *i*+1 extensions, one for each of the *i*+1 suffixes of *S* [1...*i*+1]. In extension *j* of phase *i*+1, the algorithm first finds the end of the path from the root labeled with substring *S* [*j*…*i*]. It then extends the substring by adding the character *S* (*i*+1) to its end.

Sometimes, the resulting answers are found to have irrelevant words like propositions since the Answer Extraction module finds the matching patterns regardless of the answer word type. Pattern <name> <answer> may give answer "In Pyramids" and consider "In" proposition to be an answer. Using the NER

of MADAMIRA, we check on the answers because we restrict ourselves to four types of questions: LocationCountry, LocationCity, HumanIndividual and NumericDate. MADAMIRA can find the NER of words belonging to the three major categories: LOC, PER and ORG. Other words that do not belong to these categories will not be added to the dictionary made for the word types. The answers of LocationCountry and LocationCity questions are expected to be a location. So, the system checks the words in the dictionary to make sure that these words are recognized to be entities. MADAMIRA sometimes fails in identifying location words and identifies them as person words. Our best approach is to check if these words are considered to be entities or not to be accepted as answer. In HumanIndividual questions, the answers are expected to be a person making it easy to check the words in the dictionary built to find out if it is a PER of not. The NumericDate question type is validated by checking if the word is a number or a month name.

In the proposed system, the top *five* answers are returned expecting the correct answer to be in one of them.

4 SYSTEM EVALUATION

We compare our work with previously established Arabic question answering systems, such as (Abouenour et al., 2010) and (Benajiba and Rosso, 2007b). However, each of these works have its own dataset and evaluation methods. Therefore, we build two versions of our system. The main difference between them is the Passage Retrieval module. The first system is referred to as the *online* version of the system, where Yahoo API is used for passage retrieval. The other version is referred to as the *offline* version, where the Explicit Semantic Analysis (ESA) approach is used for ranking passages retrieved from Yahoo API.

We use the standard metrics for QA evaluation, namely, *Accuracy*, *Mean Reciprocal Rank* (MRR) and *Answered* Questions (AQ). The definitions of these metrics are given below.

$$Accuracy = \frac{1}{N_s} \sum_{k \in S} V_{k,j} \quad (5)$$

$$MRR = Average_{k \in S}(\frac{1}{5} \sum_{j=1}^{5} V_{k,j}) \quad (6)$$

$$AQ = \frac{1}{N_s} \sum_{k \in S} max(V_{k,j}) \quad (7)$$

Where $V_{k,j}$ equals 1 if the answer to question k is found in the passage having the rank j, 0 otherwise.

4.1 Evaluation of the Online Version

In this version, all the proposed components of our system are used except for the module implementing the ESA. We compare the online version of our system with the system of (Abouenour et al., 2010). Their dataset of questions and answers are available. However, their obtained snippets (using Yahoo API at 2010) are not available. So, we obtain our snippets using the current version of Yahoo API. Our proposed system outperforms the work of (Abouenour et al., 2010) as shown in Table 1. For a fair comparison, given the improvements that should have been introduced to Yahoo API in the last few years, we also compare the quality of individual system components and not only the overall quality.

Table 1: System evaluation of the online version.

	Accuracy	*MRR*	*AQ*
Abouenour et al., 2010	20.20%	9.22	26.74%
Online system	26.15%	12.57	45.97%

In Table 2, the corresponding percentage of answered questions of (Abouenour et al., 2010) are listed. We choose the question types of the greatest number of questions to use for the training and testing, these types are Location, Person and Time. Moreover, these types have given the highest answered questions percentage at the work in (Abouenour et al., 2010).

Table 2: Abouenour et al. Answer Extraction Module Results.

Type	AQ
Abbreviation	0.33%
Count	0.56%
Location	4.26%
Measure	0.14%
Object	0.08%
Organization	0.33%
Other	5.24%
Person	4.83%
Time	2.33%

Using CLEF and TREC datasets, we divide the datasets into training and testing sets. Together, they are of 2,242 questions that pass through the *Answer Extraction* (AE) module. The results are shown in Table 3. They show that our AE module has a great role in improving the proposed QA system.

Table 3: Answer Extraction Evaluation for both systems.

	Abouenour et al., 2010	Online Version
Answered Questions (AQ)	15.30%	50.49%

4.2 Evaluation of the Offline Version

In this version, all the proposed components of the system are used including the ESA component. Due to time limitations, we compared our system to the work in (Abouenour et al., 2010) using only a sample of the data set, namely 200 questions and answers. The results are shown in Table 4.

Table 4: System evaluation of the offline version.

	Accuracy	*MRR*	*AQ*
Abouenour et al., 2010	20.20%	9.22	26.74%
Offline system	22.20%	8.16	47.66%

4.3 Evaluation of the Question Classifier Module

Table 5 shows the precision, recall, and *F*-Measure of the Li and Roth (Li and Roth, 2002) taxonomy when applied on Clef and Trec datasets (Abouenour et al., 2010). Table 6 shows the results of our question classifier module. Finally, Table 7 shows a comparison between Li and Roth taxonomy and our taxonomy.

Table 5: Precision, Recall and *F*-measure for the Li and Roth coarse classification.

	Precision	*Recall*	*F-measure*
Abbreviation	0.0%	0.0%	0.0%
Description	0.0%	0.0%	0.0%
Entity	26.7%	22.2%	24.2%
Human	70.8%	86.3%	77.8%
Location	78.1%	71.4%	74.6%
Number	97.8%	84.6%	90.7%

Table 6: Precision, Recall and *F*-measure for the proposed question classification.

	Precision	*Recall*	*F-measure*
Number Date	100.0%	100.0%	100.0%
Location Country	85.7%	68.6%	76.2%
Location City	74.4%	91.4%	82.1%
Human Individual	100.0%	97.1%	98.5%

Table 7: Classification results.

% of	*Li and Roth*	*Our proposed classifier*
correctly classified instances	75.1%	89.2%
incorrectly classified instances	24.8%	10.7%

5 CONCLUSIONS AND FUTURE WORK

Question Answering systems started to be a standard built-in feature in most of the search engines. However, Arabic QA systems still lag behind despite the large population of Arabic speakers. Even the latest research in this domain does not bring highly accurate results. Our aim is to increase this accuracy. A classical QA system architecture is adopted, starting with a question analysis module that mainly classifies the question and generates proper queries, adding to it a query expansion module for improving the recall of the passage retrieval module. The passage retrieval and ranking module takes a query as an input and outputs the most relevant documents to that query. Although implementing the ESA for this specific NLP problem was previously applied in (Abdelnasser et al, 2014), scaling it to fit generic questions is the challenge that we took. Our contribution shows that the ESA with proper work and computational power is giving promising results. We use a new answer extraction method that was not implemented for Arabic QA systems before. We used an NER system in the Answer Extraction module that was found to be giving much better results. The results show that; with enough data, we would produce much more pattern types giving a very acceptable output.

In the future, we want to extend our Answer Extraction module to take more question types. Moreover, we started to extend the used NER system our own named entity recognizer, as this directly affects the results returning from the answer extraction module. Finally, we are planning to create our own Arabic testbed. As opposed to Latin NLP, we do not have a single unified testbed. Finally, we would like to apply deep learning techniques and measure their contribution to the overall performance.

REFERENCES

Abdelnasser, H., Mohamed, R., Ragab, M., Mohamed, A., Farouk, B., El-Makky, N., Torki, M., 2014. Al-bayan: an Arabic question answering system for the holy

Quran. *In Arabic Natural Language Processing Workshop, page 57*, Qatar.

Abouenour, L., Bouzouba, K., Rosso, P., 2010. An evaluated semantic query expansion and structure-based approach for enhancing Arabic question/answering. *In International Journal on Information and Communication Technologies, 3(3):37–51*.

Abouenour, L., Bouzoubaa, K., Rosso, P., 2012. IDRAAQ: New Arabic question answering system based on query expansion and passage retrieval. *In CLEF (Online Working Notes/Labs/Workshop)*.

Benajiba, Y., Rosso, P., 2007a. Anersys 2.0: Conquering the NER task for the Arabic language by combining the maximum entropy with pos-tag information. *In Proc. Of Workshop on Natural Language-Independent Engineering, IICAI-2007*.

Benajiba, Y., Rosso, P., 2007b. Arabic question answering. *Diploma of advanced studies*. Technical University of Valencia, Spain.

Benajiba, Y., Rosso, P., and Benedíruiz, J. M., 2007. Anersys: An Arabic named entity recognition system based on maximum entropy. *In Computational Linguistics and Intelligent Text Processing, pages 143–153*. Springer.

Diab, M., 2009. Second generation AMIRA tools for Arabic processing: Fast and robust tokenization, POS tagging, and base phrase chunking. *In 2nd International Conference on Arabic Language Resources and Tools (Vol. 110)*.

Elkateb, S., Black, W., Vossen, P., Farwell, D., Rodríguez, H., Pease, A., Alkhalifa, M., 2006. Arabic wordnet and the challenges of Arabic. *In Proceedings of Arabic NLP/MT Conference*, London, UK.

Ezzeldin, A. M., Kholief, M. H., El-Sonbaty, Y., 2013. Alqasim: Arabic language question answer selection in machines. *In International Conference of the Cross-Language Evaluation Forum for European Languages, pages 100–103*. Springer.

Fellbaum, C., 2005. WordNet and wordnets. *In: Brown, Keith et al. (eds.) Encyclopedia of Language and Linguistics, Second Edition, pp. 665--670*. Elsevier, Oxford.

Gabrilovich, E., Markovitch, S., 2007. Computing semantic relatedness using wikipedia-based explicit semantic analysis. *In Proceedings of the 20th international joint conference on artificial intelligence, volume 6, page 12*.

Gabrilovich, E., Markovitch, S., 2009. Wikipedia-based semantic interpretation for natural language processing. *In Journal of Artificial Intelligence Research, 34:443–498*.

Habash, N., Rambow, O. and Roth, R., 2009, April. MADA+ TOKAN: A toolkit for Arabic tokenization, diacritization, morphological disambiguation, POS tagging, stemming and lemmatization. *In Proceedings of the 2nd international conference on Arabic language resources and tools (MEDAR), Cairo, Egypt (Vol. 41, p. 62)*.

Hammo, B., Abu-Salem, H., Lytinen, S., 2002. QARAB: a question answering system to support the Arabic language. *In Proceedings of the ACL-02 workshop on Computational approaches to semitic languages, SEMITIC '02, pages 1–11*. Stroudsburg, PA, USA. Association for Computational Linguistics.

Li, X., Roth, D., 2002. Learning question classifiers. *In Proceedings of the 19th International Conference on Computational Linguistics-Volume 1, pages 1–7*. Association for Computational Linguistics.

Martin, A. I., Franz, M., Roukos, S., 2001. IBM's statistical question answering system-trec-10. *In Proceedings of the 10th Text Retrieval Conference.* TREC-10.

McCandless, M., Hatcher, E., Gospodnetić, O., 2010. *Lucene in Action, Manning, 2nd Edition*.

Niles, I., Pease, A., 2003. Mapping wordnet to the sumo ontology. *In Proceedings of the IEEE International Knowledge Engineering Conference, pages 23–26*.

Pasha, A., Al-Badrashiny, M., Diab, M. T., El Kholy, A., Eskander, R., Habash, N., Pooleery, M., Rambow, O., Roth, R., 2014. MADAMIRA: A fast, comprehensive tool for morphological analysis and disambiguation of arabic. *In LREC, volume 14, pages 1094–1101*.

Ravichandran, D., Hovy, E., 2002. Learning surface text patterns for a question answering system. *In Proceedings of the 40th annual meeting on Association for Computational Linguistics, pages 41–47*. Association for Computational Linguistics.

Trigui, O., Belguith, H., Rosso, P., 2010. DefArabicQA: Arabic definition question answering system. *In Workshop on Language Resources and Human Language Technologies for Semitic Languages, 7th LREC, pages 40–45.* Valletta, Malta.

Trigui, O., Belguith, L. H., Rosso, P., Amor, H. B., Gafsaoui, B., 2012. Arabic qa4mre at clef 2012: Arabic question answering for machine reading evaluation. *In CLEF (Online Working Notes/Labs/Workshop)*.

Ukkonen, E., 1995. On-line construction of suffix trees. *In Algorithmica, 14(3):249–260*.