Using an Islamic Question and Answer Knowledge Base to answer questions about the holy Quran

Bothaina Hamoud [a] and Eric Atwell [b]

[a] Umm Al-Qura University, Mecca, Saudi Arabia
[b] School of Computing, Faculty of Engineering, University of Leeds
Leeds LS2.9JT, England
[a] Boothy2007@yahoo.com, [b] e.s.atwell@leeds.ac.uk

ABSTRACT

This paper presents the QAEQAS Quranic Arabic/English Question Answering System, which relies on a specialized search dataset corpus, and data redundancy. Our corpus is composed of questions along with their answers. The questions are phrased in many different ways in differing contexts to optimize Question Answering (QA) performance. As a complete question answering solution, the Python NLTK natural language toolkit has been used to process the user question as well as to implement the search engine to retrieve candidate results and then extract the best answer. The system takes and accepts a Natural Language (NL) question in English or Arabic from the user - through a GUI - as an input, then matches this question with the knowledge base questions, and then returns the corresponding answer. A keyword based search was used. First the user question was tokenized to get the keywords, and then the stop words were removed. The remaining keywords were used for searching the corpus looking for matched questions. After that, the system used scoring and ranking to find the best matched question and then return the corresponding answer for this question. QAEQAS deals with a wide range of question types including facts, definitions. It produces both short and long answers with a precision of 79% and a recall of 76 for Arabic version; and a precision of 75% and a recall of 73% for English version.

***Keywords*: Quran, Question, Answer, Python, Corpus, Knowledge base.**

1. INTRODUCTION

A Question Answering (QA) system is an automated approach to retrieving correct answers to questions written by users in natural language (Pujar, S. et al, 2015). A QA system is a computer program that extracts its answers from a collection of structured (database) or unstructured (text) data known as the knowledge base. The main goal of a QA system is to facilitate human-machine interactions, by getting quickly a precise answer rather than a list of documents. QA systems are classified into two categories: Closed domain and open domain QA systems. Closed-domain QA systems deal with questions in a specific domain such as Quran, medicine, or agriculture; closed-domain might point to a condition where only a limited type of questions are used, such as questions asking about description or fact. While open domain QA systems answer questions about anything in all domains and depend on general ontologies and world knowledge; moreover, open domain systems usually need much more data to extract the answer.

In general, the processing of a QA system is composed of three stages: question analysis, document analysis, and answer analysis. The question is analyzed and classified to determine

the question type and the major concept involved in the question; it is also reformulated to an appropriate query. The documents are analyzed to extract candidate documents that contain answers. The answer is analyzed to extract candidate answers and then rank to find the best one. Question answering systems integrate many techniques such as artificial intelligence, natural language processing, pattern matching, information retrieval, and information extraction. To create an accurate QA system, we need to integrate a wide range of knowledge, and use many ideas in natural language processing and artificial intelligence.

Many people do not find accurate answers for their Islamic questions even though the Quran usually provides wide and thorough teaching, not only about rituals but also about all aspects of life. Many people prefer to find their religious answers from the Internet rather than searching through classic religious books. Most search methods available through search engines cannot satisfy all users' needs to get exact and specific information. In addition, the user currently bears the burden of searching through all the documents or links provided in order to find the required answer, which is tedious and time consuming. While an automated answering system is the most important component of a distance teaching platform, it has been receiving increasingly intensive attention and is an active area for research. Different techniques and approaches have been investigated by researchers during recent years in order to provide reliable QA systems for answering online user questions. However, they do not attempt to cover questions which ask about the Quran in detail, and this poses a major religious problem for Muslims and non-Muslims. And thus, there is a need to create a QA system for the holy Quran that can retrieve small pieces of text containing the actual answer to the question posed by a user, rather than a list of documents.

The rest of this paper is organized as follows: section 2 shows some of the work related to the system. Section 3 shows the preparation of the Quran questions and answers corpus. Section 4 shows the details of the system model. Some results are shown in Section 5. In Section 6, we show the system evaluation. Finally, we conclude the paper and give directions to future work in Section 7.

2. RELATED WORK

Kenaan et al (2009) designed an Arabic QA system. The system uses an existing NLP tagger to analyze both the user's question and the documents, and uses the IR system to retrieve candidate documents containing information relevant to the user's query. The system uses keyword matching to generate the answer by matching the simple structures extracted from the question and the ones extracted from the candidate documents. Their source of knowledge is a collection of Arabic text documents. M. A.Yunus et al (2010) designed a system which uses Quran documents in three languages (Malay, English, and Arabic). It translates words of an input query into another two languages, and then retrieves verses that contain words matching the translated words. The system retrieval included some irrelevant documents which can go on to affect the system performance, and additionally it doesn't provide an actual question answering system, just a way of ranking documents according to relevance to an input search query.

L Zhenqiu, (2012) designed an automatic QA System Base on Case Based Reasoning (CBR). As the system is used, it builds up a database of previous cases of questions and answers, to guide future question-answering. The system parses the new question, extracts the keywords, and then searches the historical questions database according to the keywords to get the candidate previous questions. Then it calculates the similarity between the new questions and

candidate history questions, and according to its threshold, the system would show the answers; otherwise the questions would be recorded in the answering database as a new question. If no similarity was achieved, the system would enter the full-text search module according to the keywords of the questions in order to search the full-text.

RH Gusmita et al, (2014) developed a rule-based QA system on relevant documents of Indonesian Quran translation, using a combination of two architectures to complement each other: relevant documents, and rule-based method. H. Khan et al, (2013) suggested a Quranic semantic search tool, that worked by developing a simple domain ontology for the holy Quran based on living creatures, including animals and birds that are mentioned in the Quran. H. Abdelnasser et al, (2014) introduced an Arabic QA system for the holy Quran, which prompts the user to enter an Arabic question, then retrieves relevant verses from the Quran along with their Arabic explanations from Tafseer books, and then goes on to extract the text that contains the answer. The system integrated 2 existing ontologies to form 1,217 Quranic concepts, in addition to all the verses from the Holy Quran. They used MADA toolkit to analyze the question, and a Support Vector Machine (SVM) classifier to classify questions. The IR stage is based on the explicit semantic analysis approach. The named entities in the input question are identified by the answer extraction stage and then several features are extracted so that they can be used in the ranking of each candidate in order to extract the final answer to the input question.

3. PREPARING QURAN QUESTIONS AND ANSWERS CORPUS

A question answering system relies on a good search corpus: If documents do not contain the answer the system cannot do more. If the answer to a question is not existent in the data sources, a correct answer will not be obtained, even if the question processing, information retrieval and answer extraction models are doing well. It is thus clear that larger collection sizes of corpus normally deliver better QA performance; this can be done by integrating an enormous range of knowledge-bases. The data redundancy in huge collections means that some of the information may be phrased in many different ways in differing contexts, thus the onus on the QA system to perform complex NLP techniques to understand the text may be reduced because the right information appears in many forms. Our system differs from most question answering systems in its dependency on data redundancy rather than sophisticated linguistic analyses.

The corpus for the holy Quran is composed of a set of frequently asked questions along with their correct answers, which have been collected from several trusted expert sources, and then preprocessed. The data subsets were merged for the purpose of a Quran Q&A task. The Quran Q&A corpus collection comprises new data and some existing data from various small test-sets. Three sources have been used to collect the Quran questions and answers. The first and main source for corpus collection is a web-based tool created by a group of scholars in the Islamic field. The second source is a group from Holy Mosque in Mecca; the third is from a survey of previous research on Quran question-answering. Data preparation for a modeling tool covers all tasks to build the final dataset from the initial raw data such as table, record, and attribute selection; transformation or construction; integration; reformatting; and data cleaning. Data preparation tasks can be performed multiple times and not in a specific order (Hamoud and Atwell, 2016).

4. SYSTEM MODULE

4.1 General description

In our prototype version to answer questions from the holy Quran, a single source of data has been used as knowledge base for Arabic and English. Keyword-based techniques have been applied to return an answer. The general idea of QAEQAS is: The system takes and accepts a Natural Language (NL) question from the user as an input, matches this question with the questions found in the data set, and then returns an answer for the user question as an output. The answers may be short phrases or longer answers, like sentences, or even many paragraphs, to provide for question clarification. As a complete question answering solution, the Python NLTK Natural Language Toolkit was used to process the user question as well as to work as search engine to retrieve candidate results and finally extract the answer. While the context of the input question can differ from user to user, many of the questions were phrased in many different ways in differing contexts. Data redundancy in large collections makes it easier for the system to find the right information. QAEQAS deals with a wide range of question types including: facts, definitions, how, why, and hypothetical. Furthermore QAEQAS accepts both short and long questions. The QA Pairs has been converted into the appropriate comma-separated value (CSV) format which is convenient and suitable for our question answering system.

4.2 Preparing the knowledge base file

Some questions along with their answers have been selected from the collected dataset, which were used as examples for the model. This dataset is composed of 500 questions and answers for the Arabic version and 350 for the English version. Since the Excel application produces and uses CSV files, these questions and their answers are entered in one sheet of an Excel workbook file, so that each question and its corresponding answer is a record (row). The table consists of two column with heading “question” and “answer”. After that the excel file has been saved as CSV file format.

4.3 Building a graphical user interface

Python tkinter module was used for creating a graphical user interface (GUI) that facilitates user interaction with QAEQAS. The GUI allows a user to enter his/her question in natural language, and accepts the question. After the system processes the question and searches for the answer, the GUI accepts the returned answer if there is any, or a message telling the user that there is no answer for the question. Python has more than one GUI package, but tkinter is the standard and the most commonly used one. The tkinter module (Tk interface) is also the standard Python interface to the toolkit GUI from scriptics. tkinter comprises of a number of modules. The Tk interface is provided by a binary extension module named _tkinter. The public interface is provided through a number of Python modules. To use tkinter, we should import the tkinter module using: import tkinter, or, from Tkinter import * (Lundh, 1999).

4.4 Tokenizing the user question

Tokenization is the process of segmenting a text into linguistic units such as words, punctuation, numbers, alphanumeric, symbols, phrases, paragraphs, sentences or any other expressive elements known as tokens. This tokenized text is used as an input for further processing. Python Natural Language Tool Kit NLTK provides a number of tokenizers in the

tokenize module, which are used to split strings into lists of substrings (Bird et al, 2009). They provide several ways to tokenize text: word_tokenize tokenizer is used to detect words and punctuation in a text, it needs the Punkt sentence tokenization models to be installed. NLTK also has a simple tokenizer known as wordpunct_tokenize, used to divide text on whitespace and punctuation. wordpunct_tokenize is based on regular expressions. Tokenization can also work at the level of sentences. Using the sentence tokenizer by importing sent_tokenize, word_tokenize to divide the text to sentences and then to words (Perkins, 2014).

One extra problem with Arabic questions and answers is that these tokenizers work on decoded version of a Unicode string only and not encoded version, therefore with Arabic text the string should be decoded first. To decode a string in Python we use the s.decode("utf8"). The word_tokenizer(s) has been used to split the user question into tokens composed of words and punctuation which are used in searching the corpus questions.
Removing the stop words from the user question.

Stop words are the most common words in any language such as ("the", "a", or "and") in English language, which help build ideas, for example, linking sentences or words, but do not give any meaning. These words probably appear in many questions, so they are often separated out and removed before or after the processing of natural language data (text). There is no specific standard list of stop words that is used by all natural language processing tools. In fact not all tools use a stop words list. Some tools do not remove these stop words to help in phrase searching (Ullman et al. 2011). Any collection of words can be selected as the stop words for a specific purpose; it is not easy to determine the stop words, and on the other hand, stop words differ according to the case used.

For some English-language search engines, these are some of the most common, short function words, such as "the, is, at, which, and on". In this situation, stop words may cause problems when searching for sentences or lines that include them, especially in names. Other search engines remove some of the most common words —including lexical words, such as "want"— from a query in order to improve performance. The English stop words list that included within NLTK has been used to remove all stop words that are found in the user question. For Arabic, stop words list are not included in NLTK; hence we used a version of Arabic stop words created by Taha Zerrouki (Zerrouki et al 2014).

4.5 Searching for answer to the user question in the Quran Q&A corpus

To find matches we work with regular expressions in Python. The "re" package is used to carry out queries on an input text file. The '.re' package has several functions such as re.match(), re.search() and re.findall(). Each of these functions takes a regular expression, and string to find matches. The match function only finds matches if they exist at the beginning of the string. The re.search function scans forward through string looking for a position where the regular expression pattern finds a match, and then returns a corresponding matched result. Both functions return "None" if no match can be found. The re.search() function has been used to search the CSV file looking for questions that match the keywords produced from user question, and then return the candidate matched questions. The regular expression "re.I" has been used to enable a case-insensitive mode (ignoring case), for example, 'surah' will match the string 'Surah'. The characters ^ and $ can be used for the beginning and the end of the question.

When the corpus has been searched without removing the stop words included in the user question, too many matches were found. After removing the stop words, there were fewer answers displayed than before. After that this group of stop words has been studied and examined against the corpus, in both English and Arabic, and we realized that there are some words that appear in many questions of the corpus, but were not included within the standard stop words lists. Some of these words are used to help in building questions such as the Arabic words: (رأي, وصف ذكر, ورد,), and English words (mention, describe etc.). Other words are considered as common words in Quran domain or might be used frequently in Islamic literature such as Arabic words (تعالى, سبحانه, الكريم, (ص), عزوجل), and English words (Peace, upon, Sallallahu, Alaihi, Wasallam, holy, etc.). We realized that these words should be added to the stop words lists to improve the search performance.

4.6 Scoring and Ranking the results

A ranking is a relation between a set of items such that the items are organized in ascending or descending order according to specific criteria. Using rankings, information can be easily evaluated according to a particular norm, for example, the search engine ranks the pages according to their relevance to the user query, which makes it simpler for the user to select the pages they want. As it has been seen before, QAEQAS sometimes displays a collection of candidate results, but we don't know which the best ones are. One possibility is to use a similarity measure to rank the candidates. One good feature is the degree of match between the user question and returned result. The score for the results that are returned after searching can be used for this purpose. So a scoring method has to be applied for the candidate results and then ranked them according to their high score. A score was then given to each of these candidates according to the number of question words it contains, the more the better. Assigning scores represents the degree of results relevance in respect to the user question. It has been noticed that sometimes QAEQAS displayed a wrong answer, because the same word may appear more than once in the candidate and gives it increased scores, or some candidates have the same score. The first problem has been handled by counting the repeated word only once in the score. After that the results has been ranked in descending order according to their score.

4.7 Displaying the answer

The answer extraction process relies on finding the correct question from the knowledge base, that matches the user question, and then finding the complete line which is composed of the question and the answer. To obtain the answer Python functions have been used to divide the line into two to get the answer part, and then display it.

5. RESULTS AND DISCUSSION

Searching for answers in questions & answers corpus (knowledge base) without tokenizing the user's question gives no relevant result as shown in Fig.1, because the system searches for the whole question. It gives an answer only when the user question is a full exact match to a question found in the corpus, and that may happened rarely.

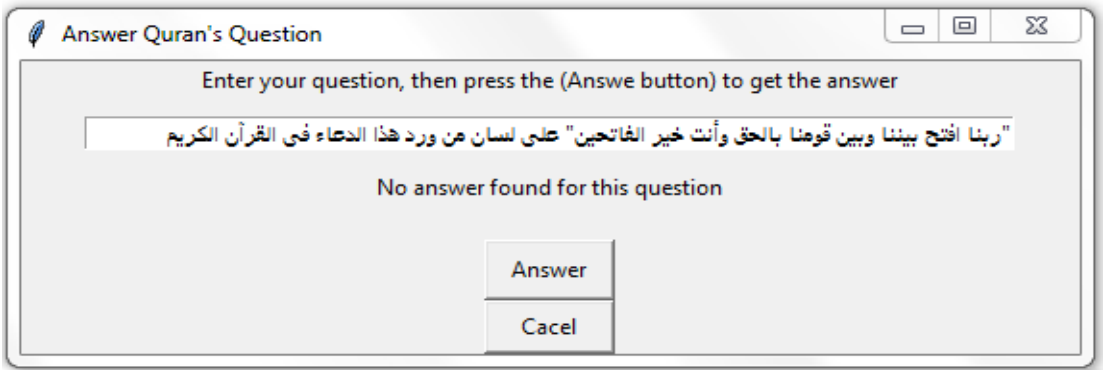

Fig. 1: Search result before tokenizing the user query

After tokenizing the user query, many answers have been displayed, because the system searches for matches in the corpus questions for every keyword found in the user question. After removing standard stop words the numbers of answers became less than before. When adding some words to the standard stop words and removing other stop words, the system displayed even fewer answers; Fig. 2 shows the candidates result for the user' question "Who is the relative of the Prophet Muhammad Peace be upon him, whose name is mentioned in the Quran?", after removing all stop words.

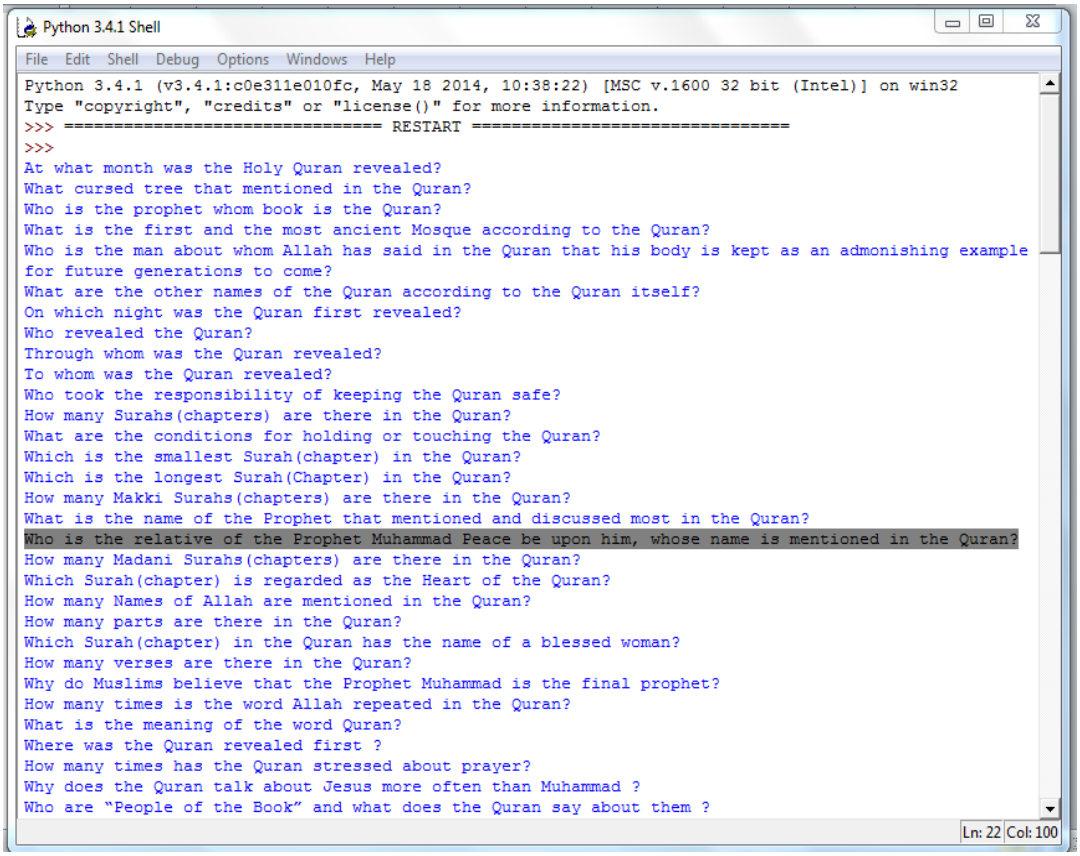

Fig. 2: Search result after tokenizing and removing the stop words from the user's query

Furthermore, it has been found that more relevant results were achieved after scoring and ranking the candidates. Fig. 3 shows the search result after scoring and ranking - to find the best answer- which significantly improves the search performance. It has been realized that the system produces wrong answers, when the user's question is short i.e. composed of few keywords, or when the context of the user question is totally different from the context of the question found in the corpus.

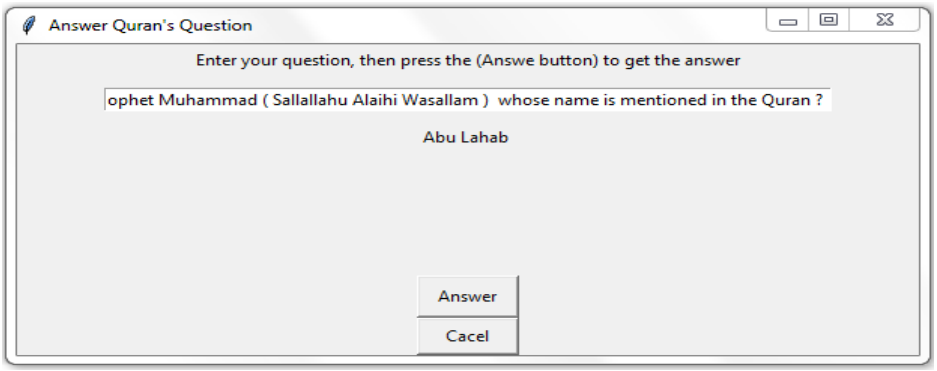

Fig. 3: Search result after tokenizing, removing the stop words, scoring and ranking

6. SYSTEM EVALUATION

Characteristics of QAEQAS have been investigated through user-oriented testing. Evaluating QAEQAS is based on domain expert knowledge, since the corpus is composed of pairs of questions, and its correct answers from credible sources. Since we do not have a Gold Standard for the Quran questions to compare with the results, we relied on an expert user group of Muslim university academics to ask QAEQAS some questions in our question domain. Table 1 details the evaluation for Arabic and English questions with the answers retrieved by the system. Table 2 shows some of Arabic questions that were incorrectly answered by the system.

Table 1: Evaluation of Arabic and English answers retrieved by the system

Number of questions	**Arabic**	**English**
Total number of questions	71	63
number of correctly answered questions	54	46
number of incorrectly answered questions	14	15
number of answers the system failing to return	3	2
total number returned answer	68	61

Accuracy for the Arabic version:

Precision = rlv / ra → 54/68 = 79%

Recall = rlv/ ta → 54/71 = 76%

Accuracy for the English version:

Precision = rlv / ra → 46/61 = 75%

Recall = rlv/ ta → 46/63 = 73%

rlv = the number of correctly answered questions (relevant)

urlv = the number of incorrectly answered questions (irrelevant)

ra = the total number returned answer (rlv+urlv)

urta = the number of answers the system failing to return (unreturned)

ta = the total number of questions (returned + unreturned)

Table 2: Examples of Arabic questions that were incorrectly answered by the system

User question	Corpus question	Result &discussion
من هو النبى الذى ذكر فى القرآن باسم اسرائيل ؟	كل الطعام كان حِلاً لبني إسرائيل إلا ما حرَّم إسرائيل على نفسه, فمن هو إسرائيل ؟	The user's question and the corpus question are different in context
ذكر اسم صحابى من صحابة الرسول صلى الله عليه وسلم في القران الكريم فمن هو؟	من هو الصحابى الذى ذكر اسمه صراحة فى القرآن ؟	The words "القران" , "الله" , "الرسول" are found in many questions, and one of these questions gets higher score than the intended question.
ما اقسام الانفس في القرآن	ما أنواع الأنفس التى ذكرت فى القرآن الكريم	More than one question has the same score , the system displays the first one from the tied score
ماهي اوهن البيوت المذكوره في القرآن	ما أضعف بيت ذكر فى القرآن الكريم	There is a question in the corpus gets higher score than the intended one

7. CONCLUSION AND FUTURE WORK

This paper presents QAEQAS, the Quranic Arabic/English Question Answering System that deals with a wide range of question types including: facts, lists, definitions, and hypothetical questions, with a precision of 79% and a recall of 76 for Arabic version; and a precision of 75% and a recall of 73% for English version. We are currently working on a semantic model of question understanding and processing, by finding keywords in a query, and determining the contextual meaning of the words that the user intended, to supply the user with more meaningful search results after evaluating and understanding the search text, and finding the most relevant results in the data set. The system performance may be further improved by applying semantic search techniques. A lexical semantic ontology such as WordNet can then be used for understanding the context. Furthermore the searched text can also be analyzed.

8. 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. Proceedings of the EMNLP 2014 Workshop on Arabic Natural Langauge Processing (ANLP), pages 57–64, Doja, Qatar.

Bird, S., Klein, E., & Loper, E. (2009). Natural language processing with Python. O'Reilly Media, Inc.

Comma Separated Values (CSV) Standard File Format. http://edoceo.com/utilitas/csv-file-format

Gusmita, R. H., Durachman, Y., Harun, S., Firmansyah, A. F., Sukmana, H. T., & Suhaimi, A. (2014). A rule-based question answering system on relevant documents of Indonesian Quran Translation. Proceedings of International Conference on Cyber and IT Service Management (CITSM). pp. 104-107.

Hamoud, B., & Atwell, E. (2016). Quran question and answer corpus for data mining with WEKA. In the Conference of Basic Sciences and Engineering Studies (SGCAC). pp. 211-216. IEEE.

Kanaan, G., Hammouri, A., Al-Shalabi, R., & Swalha, M. (2009). A new question answering system for the Arabic language. American Journal of Applied Sciences, 6(4), pp.797-805.

Khan, H. U., Saqlain, S. M., Shoaib, M., & Sher, M. (2013). Ontology based semantic search in Holy Quran. International Journal of Future Computer and Communication, 2(6), pp570-575.

Lundh, F. (1999). An introduction to tkinter. http://www. pythonware.com/library/tkinter/introduction/index.htm.

Perkins, J. (2014). Python 3 Text Processing with NLTK 3 Cookbook. Packt Publishing Ltd.

Pujar, S., Priyaa, B., & Sethia, K. (2015). Distributed QA System. Research Report, New York University, USA.

Shawar, B. A., & Atwell, E. (2009). Arabic question-answering via instance based learning from an FAQ corpus. In Proceedings of International Conference on Corpus Linguistics. Lancaster University, UK.

Sherkat, E., & Farhoodi, M. (2014). A hybrid approach for question classification in Persian automatic question answering systems. In the 4th International eConference on Computer and Knowledge Engineering ICCKE. pp. 279-284.

Ullman, J. D., Leskovec, J., & Rajaraman, A. (2011). Mining of Massive Datasets

Yunus, M. A., Zainuddin, R., & Abdullah, N. (2010). Semantic query for quran documents results. In IEEE Conference on Open Systems (ICOS) pp. 1-5. IEEE.

Zhenqiu, L. (2012). Design of automatic question answering system base on CBR. Procedia Engineering, 29, 981-985.

Zerrouki, T., Alhawait, K., & Balla, A. (2014). Autocorrection Of Arabic Common Errors For Large Text Corpus. Proceedings of the EMNLP 2014 Workshop on Arabic Natural Langauge Processing (ANLP), pages 127–131, Doha, Qatar.