JURNAL INFOTEL
Informatics - Telecommunication - Electronics

Website Jurnal : http://ejournal.st3telkom.ac.id/index.php/infotel
ISSN : 2085-3688; e-ISSN : 2460-0997

Question Answering Machine for Islamic Factual Questions Based on English Translation of Hadith

Ja'far, Adiwijaya[2], Said Al-Faraby[3]
[1] Fakultas Informatika, Universitas Telkom, Bandung
[2] Divisi Digital Service PT Telekomunikasi Indonesia
[1]Bandung, West Java, Indonesia
*jafarassagaf@students.telkomuniversity.ac.id, adiwijaya@telkomuniversity.ac.id, saidalfaraby@telkomuniversity.ac.id

Received, Revised, Accepted

Abstract — The basic idea of this project is to create a system that can answer Islamic factual question. With English translation of Hadith Sahih Bukhari book as the corpus for answers. The corpus is not a knowledge base, it is a document consisting 7095 verses of Hadith. Hence, it is inherently harder for the system to read and extract answer directly. The system able to receive an input posed in natural language such as "When was Prophert Muhammad born?" or "What are the best deed we can do to pleases Allah?". A keyword based search was used. First the user question went through preprocessing phase where keyword is extracted. Then, the keywords were used to search the corpus looking for matching verses. After that, the system used scoring and ranking to find the best matched verse and then return the corresponding answer for the question.

Keywords – system, question, answer, hadith, islam.

I. INTRODUCTION

A Question Answering (QA) system is an automated approach to retrieving correct answers to question written by user in natural language. A QA system is a computer program that extracts its answers from a collection of structured (database) or unstructured (text) data. The main goal if 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 question asking about description or fact. while open domain QA systems answer questions about anything in all domains and depend on general ontology and world knowledge; moreover, open domain systems usually need much more data to extract the answer [1] . 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(AI), natural language processing(NLP), machine learning, 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.

Knowledge about Islam is a mandatory subject that all of us Muslim should be aware of. But sometimes, be it young or old people, basic information about Islam such as the number of tahajud prayer rakaah, the date of isra mi'raj, or the year of which our prophet hijra to Medina are all easily forgotten. For some, it's easy to obtain/regain the information by reading the Holy Quran or a collection of hadith, but for the people that have time off the essence but wanted to get answers directly from Book of hadith, sitting down and

searching the whole Books for a specific piece of information is regarded not efficient. Question Answering (QA) machine is one of many alternative to solve this problem. With QA Machine, human need not to do extra work to find the answer the are looking for. For instance, when someone is curious about which country plays a great role in the beginning of world war two, they can ask Google assistant to extract the information instead of reading a book of world war two for one and a half hours, thus saving a lot of time. This is how the methods works ; The system takes a natural language question as an input rather than a set of keywords, for example, "When was our prophet Muhammad born?". The sentence is then classified and transformed into a query through its logical form. Having the input in the form of a natural language question makes the system more user-friendly, but harder to implement. As there are various question types and the system will have to determine the correct one in order to give sensible answer. Assigning a question type to the question is a crucial task, the entire extraction process relies on determining the correct question type and hence the correct answer type.

The QA is a closed domain system, meaning that only a specific topic and type of question are dealt with. In particular, it only answers Islamic question that require facts as the answer. Book of hadith By Sahih Bukhari with 7095 Hadiths was used as the corpus. The input is a question posed in natural language and should be related to Islamic matters and posed with proper English grammar. Question Answering Machine for Islamic Factual Questions Based on English Translation of Hadith is a machine that was designed to build a system that can answer Islamic related question posed by user in natural language, determine whether Book of Hadith by Sahih-bukhari is a viable corpus for a QA system, and help one who desire to obtain or regain information regarding Islam knowledge\\

II. RESEARCH METHOD

A Machine Called START (SynTatic Analysis using Reversible Transformations) by Boris Katz [2] at MIT's Artificial Intelligence Laboratory had been designed to answer question that are posed in natural language by users. START Parse incoming questions, match the queries created from the parse trees against its knowledge base and presents the appropriate information segments to the use. Int this way, START provides untrained users with speedy access to knowledge that in many cases would take an expert some time to find [3].

Searching the Quran for keywords using one English translation produced incomplete results. The recall values improved when the number of translations used in the search process increased as in the case of Quran Iman search tool (Uses five English translation by default). As the file that contains the eight English parallel translations is available as an open source resource on the web, it was used to improve the recall values even further reaching a value of 87 percent [4]. A Study by Noorhan Hassan Abbas which is Quran "Search for a Concept" [5]. Tool and website had allowed the user to search in Holy Quran by two-way first one search by keyword and by the concept. She generated a tree view concept for Quran based on "Mushaf Al Tajweed" Ontology . In 2007 a chat bot was reported being developed for the Quran at Leeds University; this chat bot answers questions from the Quran but it does not really understand the question. It just tries to look for the most 'impactful' words in the input, and then retrieve verses from the Quran that is related to these words.

L Zhenqiu [6] 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 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. Kanan et al [7] designed an Arabic QA system. The system uses an existing NLP tagger to analyze both 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 an the ones extracted from the candidate documents. Their source of knowledge is a collection of Arabic text documents. H. Abdelnasser et al [8] introduced an Arabic QA system for the holy Quran, which propmts 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 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 an 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.

Noorhidawati [9] designed a system which uses Quran in three languages (English,Arabic,Malay). 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 document according to relevance to an input search query. RH Gusmita et al [10] 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.

Elasticsearch is a highly scalable open-source full-text search and analytics engine. It allows you to store, search, and analyze big volumes of data quickly and in near real time. It is generally used as the underlying engine/technology that powers applications that have complex search features and requirements [11]

III. QNA MACHINE

This part describes the process that are applied in bulding the Qna Machine. From scrapping hadith from the internet, cleaning the hadith data-set from noises, uploading the offline hadith data-set into elasticsearch, classifying question, extracting features from the input,constructing the query, searching for candidate answer, and constructing the answer.

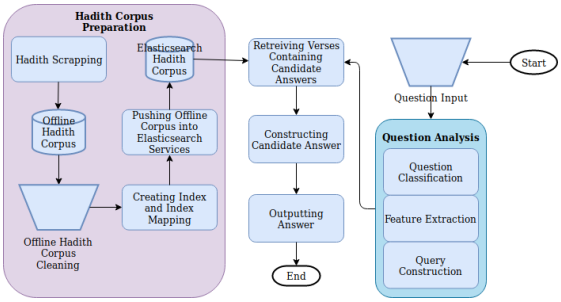

Figure 1 System Architecture

A. *Hadith Scrapping*

To build the scrapper first we need to decide which website provides the most suitable data set. In this case sahih-bukhari.com provides all the variables needed. The site-map let us find all the URL that leads to each books of hadith. The scrapper accessed all the URL provided and then find all the hadith using regular expression. There are a total of 93 pages that needs to be accessed. The hadith consist of five headers, ‘Volume’, ‘Book’, ‘Number’, ‘Narrator’, and ‘Ayah’. Each of them contains a string values. The program access all the 93 pages to find all the hadith.

The Hadith Scrapper was built using Python language with the help the help of urllib module to access the website.

```
<div align="left" class="Style9"><strong>Volume 1</strong></div>
<p align="left"> <a href="Pages/Bukhari_1_01.php" class="Style8">1. Revelation</a><br />
      <a href="Pages/Bukhari_1_02.php" class="Style8">2. Belief</a><br />
      <a href="Pages/Bukhari_1_03.php" class="Style8">3. Knowledge</a><br />
      <a href="Pages/Bukhari_1_04.php" class="Style8">4. Ablution (Wudu')</a><br />
      <a href="Pages/Bukhari_1_05.php" class="Style8">5. Bathing (Ghusl)</a><br />
      <a href="Pages/Bukhari_1_06.php" class="Style8">6. Menstrual Periods</a><br />
      <a href="Pages/Bukhari_1_07.php" class="Style8">7. Ablution with dust</a><br />
      <a href="Pages/Bukhari_1_08.php" class="Style8">8. Prayer (Salat)</a><br />
      <a href="Pages/Bukhari_1_09.php" class="Style8">9. Prayer Hall (Sutra)</a><br />
      <a href="Pages/Bukhari_1_10.php" class="Style8">10. Times of the Prayer</a><br />
      <a href="Pages/Bukhari_1_11.php" class="Style8">11. Call to Prayer</a><br />
    <a href="Pages/Bukhari_1_12.php" class="Style8">12. Characteristics of Prayer</a></p>
<div align="left" class="Style9"><strong>Volume 2</strong></div>
<p align="left"><a href="Pages/Bukhari_2_13.php" class="Style8">13. Friday Prayer</a><br />
      <a href="Pages/Bukhari_2_14.php" class="Style8">14. Fear Prayer</a><br />
      <a href="Pages/Bukhari_2_15.php" class="Style8">15. The Two Festivals (Eids)</a><br />
      <a href="Pages/Bukhari_2_16.php" class="Style8">16. Witr Prayer</a><br />
```

Figure 2 sahih-bukhari.com sitemap

After all the hadith had been scrapped as a csv file, the data need to be manually cleaned using spreadsheet replace feature from all Unicode noise such as the excessive amount of "/" character .

B. *Building Document-Oriented Database using Elasticsearch*

After the offline hadith data-set has been cleaned, the data-set need to be saved in elasticsearch online database for searching purpose later. We use elasticsearch because elasticsearch provides automatic indexing for the document. Elasticsearch also provides near real-time search engine that can be scaled and customized to fulfill our needs.

Figure 3 Index Mapping

To build the database, first we need to create a python file to connect our machine to the elasticsearch services. Then, we create an index mapping and node setting for our hadith data-set. After the mapping has been set up, we use Bulk API to push our 7095 documents into our node.

C. *Input Classifying*

We classify the question to the anticipated type of answer. This information would help to determine whether the system is able to understand what is being questioned. In this project we build a SVM classifier to classify questions in which the question word is omitted. For example the questions: (Where did Al-Aqsa mosque was built?) and (What is the name of the place that Prophet Muhammad receive his first revelation?) both have the same question type (Location). There are two phase of classifying the input. First, we build a SVM using linearSVC (supervised vector classifier) trainer that is able construct a trainer model pickle file. Then we build the classifier for user inputted query by utilizing the trainer model we build previously. The implementations is

based on liblinear. The fit time complexity is more than quadratic with the number of samples which makes it hard to scale to dataset with more than a couple of 10000 samples. We use liblinear instead of libsvm because it has more flexibility in the choice of penalties and loss functions and should scale better (to large numbers of samples). The multiclass support is handled according to a one-vs-one scheme.

D. Features Extraction and Query Construction

Input preprocessing is done by applying morphological analysis to identify the structure of text such as: lemmas, roots, affixes, stems, part of speech(POS) tags, etc. For preprocessing we used Spacy module which is one of the most accurate english preprocessing toolkits. Spacy provides integrated word vectors and a fast |\& accurate part of speech (POS) tagger with dependency parser. In this process, first the question details (Lemma, tag, Entity Type, dependency, and word parent) is extracted. Then for each word in the question sentence, the system will determine whether the word is a noun or proper noun (be it singular or plural). If the noun is not a compound noun, then the system will recursively find the left and right word for compound noun. Then the system will fetch all the adjectives describing the noun. If there are a parent word with (ROOT) as dependency, all child word with (acomp, xcomp, advmod, and ccomp) as dependency will also be extracted as features.

After all the features has been extracted, the system will start to construct query by finding any conjunction, negation or marker words that can alter the meaning of the question. The query is constructed to show relationship amongst different features in the feature set. A conjunction is a part of speech that is used to connect words, phrases, clauses or sentence. We treat conjunctions asymmetrically, meaning that the head of the relation is the first conjunct and all the other conjuncts depend on it via the conjunction relation. A Conjunct is the relation between two elements connected by a coordinating conjunction, such as and, or, etc.

To eliminate the problem of synonyms words, we applied Query Expansion technique using WordNet. The query expansion module currently does not exist in the whole question processing pipeline. It's only for giving suggestion for alternate query if the outputted answer does not satisfy user.

E. Searching and Constructing Candidate Answer

Searching for candidate answer relies on proper question features and good corpus. If documents does 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 clear that larger collection size of corpus normally deliver better QA performance. The data redundancy in huge collections means that some of the information may be phrased in many different ways in differing context, thus making the system ability to extract answers reduced because the right information appears in many forms.

The process of searching and constructing candidate answer is divided into two parts. The first one is using Elasticsearch to retrieve ten relevant document(verses). And the second process is extracting and constructing sentences provided by previous process. The first part work like this, first, Elasticsearch finds all the appropriate documents based on queries provided. This means it receives a Boolean response of 0 if the document is not suitable, and 1 if the document is suitable. Next, for all documents with the response equal to 1, the scoring will be calculated and they will be sorted by this value. The value of scoring is a complex concept, but in general, when requesting matches with a document, more request matches result in a higher scoring. But even full match does not guarantee that the document with the highest scoring is exactly what we are looking for. For example, if the user enters "Pillars," we may be looking for information about five pillars of Islam, and not about the pillars that suspend a mosque. Just getting a match to one or more terms in a document field does not equate to relevance. Likewise, just because we didn't get a match, doesn't mean the document isn't relevant. Relevance is the numerical output of an algorithm that determines which documents are most textually similar to the query.

Elasticsearch enhances standard scoring algorithms and encapsulates these within script score and function score. These two factors combine into a calculation of the weight of a single term in a particular document. By default, Elasticsearch makes use of the Lucene scoring formula, which represents the relevance score of each document with a positive floating-point number known as the score. A higher score results in a higher relevance of the document. A query clause generates a score for each document, and the calculation of that score depends on the type of query clause. Relevance is calculated using the formula in figure 4 [12].

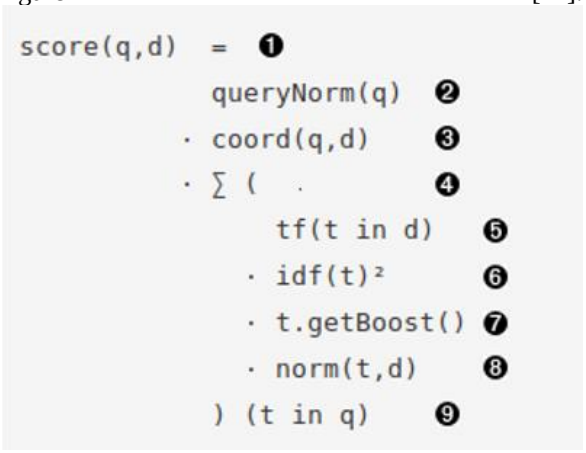

Figure 4 Relevance Formula

a) score(q,d) is the relevance score of document d for query q
b) queryNomr(q) is the query normalization factor
c) coord(q,d) is the coordination factor
d) The sum of the weights for each term t in the query q for document d.
e) tf(t in d) is the term frequency for term t in document d.
f) idf(t) is the inverse document frequency for term t.
g) t.getBoost() is the boost that has been applied to the query
h) norm(t,d) is the field-length norm, combined with the index-time field-level boost, if any.

After the top ten verses had been retrieved, each of them goes through the second part of the process. In this part, firstly the document is transformed into vector in the form of bag of words(BOW). Then, we truncate the vector matrix into singular valued decomposition using gensim lsi module so that we are able to compare the retrieved documents with the lsi model of the query. After the matrix has been truncated, we compute the similarity against inputted query by storing the sparse index matrix in memory. The similarity measure used is cosine between two vectors. After that, we sorted the document from the most similar to least similar. The top five most similar document then get extracted as the answer.

IV. RESULT AND ANALYSIS

Evaluating overall system performance is not an easy task. We do not have a definitive standard for the Islamic question to compare with our results. Human judgments can be considered a gold standard because human have the ability to judge the semantic relatedness of texts. Therefore, we asked some experts in Islam to judge our system accuracy. The system was evaluated by two Habib using a set of question. The output of our system for each question was the top-5 answers and the top-10 related verses. Each expert marked each verse or answer as right or wrong.

We did an evaluation to all our module and also overall system performance. Overall system performance have been analyzed through user-oriented testing. Evaluating the system is based on domain expert knowledge, since the corpus is composed of pure hadith verses. Table 1 details the evaluation for English questions with the answers retrieved by the system. TopN accuracy is used to evaluate the overall system. TopN accuracy of correct answers is calculated as the number of questions in which at least one of the top N answer candidates is correct, divided by the total number of questions. We also calculate the verses retrieval module using Top-1 and Top-5 accuracy. In overall system evaluation, only precise answer is counted. Other relevant document that was retrieved but does not comply as an answer were not included in the calculation. Table 2 shows the result from overall system.

Table 1 Evaluation of Answers Retrieved by System

Output	Total Answers
Correctly retrieved and answered	21
Right verses retrieved, but wrong sentences	9
Wrong verses, but still relevant	5
Wrong verses retrieved, and not relevant	5

Table 2 Overall System Accuracy

	Overall system
Top-1	30%
Top-5	52%

Low overall performance score is caused by several case. The shape of the corpus is a collection of a hadith verses, not a knowledge base(KBs). Retrieving answers directly from hadith document than from KBs is harder because information is far less structured, is indirectly and ambiguously expressed, and usually scattered across multiple documents. This condition makes it harder for the system to extract exact answer because the system relies on keyword searching method. Keyword searching works by matching input with the corpus hoping to find relevant document. When the system found a matching document, it does not concern whether the document had the same context with the question, as long as it contains matching keywords. Furthermore, implementing keyword search method into a hadith document does not always bore a result because sometimes the words in the question does not exist in the document corpus.

In evaluating IR module, verses retrieved that does not contain precise answer is also counted as long as it is still relevant with the input. We use Top-1 and Top-5 accuracy to evaluate the IR module. We also calculate the precision when the system outputs five verses for each question. The precision is calculated using the following formula :

precision = rlv/ra ---D 175/200 = 0.875

rlv = the number of correctly retrieved documents (relevant documents)

ra = the total number of returned documents\\

Table 3 shows the accuracy of IR module.

Table 3 IR Module Performance

	IR Module
Top-1	62.5%
Top-5	87.5%
Precision	87.5%

We find it quite challenging to devise the best feature model for our query because English language have a quite complicated language structure. For some

cases our model cannot extract the proper features because of how the question is structurally posed. For example, the question "What does prophet Muhammad do when eclipse happen?" does not detect the word "eclipse" as a keyword feature, thus making the system not looking for "eclipse" in the IR module. Whereas the question "what does prophet Muhammad do when eclipse occurred?" return "eclipse" as one of the keyword features. A strict feature extraction model will hinder some of the auxiliary verb that might be important for the IR process, whereas a more lenient feature extraction model might include unnecessary words that will broaden the search scope. Furthermore, using only one English translation ward the possibilities of matching semantically same word. The words "Allah's Apostle", "Allah Messenger", and "Prophet" have the same meaning, but the current system would not be able to understand that. Applying Expansion handle this problem partially.

V. CONCLUSSION

We proposed a Question Answering for English translation of Al-Hadith that takes English question as input in the form of Natural Language and retrieve relevant verses as candidate answers. Our initial result shows that using Al-Hadith document as a corpus for Question Answering Machine is feasible despite the low overall performance. To improve the overall performance of the system, in the future we propose to implement question and corpus classification using conceptual ontology to narrow down the search scope. We also plan to include active learning techniques which is convenient when gold standard is hard to obtain. Hadith experts can give their feedback and the system will learn it's pattern, thus improving the performance of future Question Answering Machine. Furthermore, we believe using parallel English translation of Al-Hadith will benefit us to improve the recall value of the system.

VI. REFERENCES

[1] B. Hamoud and E. Atwell, "Using an Islamic Question and Answer Knowledge Base to answer questions about the holy Quran," *International Journal on Islamic Applications in Computer Science And Technology*, vol. 4, no. 4, 2017.

[2] B. Katz, G. Borchardt and S. Felshin, "Natural Language Annotation for Question Answering," *Computer Assisted Information Searching on the Internet*, 1997.

[3] B. Katz, "How START works," MIT, 1990. [Online]. Available: http://start.csail.mit.edu/start-system.php. [Accessed 12 January 2019].

[4] H. SMO and E. Atwell, "Concept Search Tool for Multilingual Hadith Corpus," *International Journal of Science and Research (IJSR)*, pp. 1326--1328, 2016.

[5] N. Abbas, "Quran 'Search for a Concept' Tool and Website," 2009.

[6] L. Zenqiu, "Design of Automatic Question Answering System Base on CBR," *Procedia Engineering*, pp. 981 - 985, 2012.

[7] G. H. Kanan, A. Al-Shalbi and S. Majdi, "A New Question Answering System for the Arabic Language," *American Journal of Applied Sciences*, vol. 6, 2009.

[8] H. Abdelnasser, M. Ragab, M. Reham, M. Alaa, F. Bassant, N. M. El-Makky and M. Torki, "Al-Bayan: An Arabic Question Answering System for the Holy Quran," 2014.

[9] M. Yunus, M. Amin, Z. R and A. Noorhidawati, "Semantic query for Quran Documents Results," *ICOS 2010 - 2010 IEEE Conference on Open Systems*, pp. 1 - 5, 2011.

[10] G. H. R, Y. Durachman, S. Harun, F. A.F, S. H.T and S. A, "A rule-based question answering system on relevant documents of Indonesian Quran Translation," *2014 International Conference on Cyber and IT Service Management (CITSM)*, pp. 104-107, 2014.

[11] B. Katz, "Elasticsearch Getting Started," Elasticsearch, 2019. [Online]. Available: https://www.elastic.co/guide/en/elasticsearch/reference/current/getting-started.html. [Accessed 14 May 2019].

[12] L. p. guide, "Lucene's Practical Scoring Function," Lucene, 2019. [Online]. Available: https://www.elastic.co/guide/en/elasticsearch/guide/current/practical-scoring-function.html. [Accessed 14 May 2019].