Journal of Fundamental and Applied Sciences

ISSN 1112-9867

Available online at http://www.jfas.info

Research Article

Special Issue

# RESOURCE DESCRIPTION FRAMEWORK TRIPLES ENTITY FORMATIONS USING STATISTICAL LANGUAGE MODEL

R. A. Kadir[1,*] and R. A. Yauri[2]

[1]Institute of Informatics Visual, Universiti Kebangsaan Malaysia, 43600 Bangi, Selangor, Malaysia

[2]Department of Computer Science, Kebbi State University of Science and Technology, Nigeria

Published online: 05 October 2017

## ABSTRACT

A method in formatting unstructured sentences from the source corpus to a specific knowledge representation such as RDF is needed. A method for RDF entity formations from a paragraph of text using statistical language model based on N-gram is introduced. The implementation of RDF entity formation is applied on natural language query for information retrieval of the Islamic knowledge. 300 concepts from the English translation of Holy Quran with 350 relationships are used as a knowledge base. We evaluate our approach on collection of queries from the Islamic Research Foundation website with a total, 82 queries and compare the performance against previous method used in FREyA. The result shown the proposed method improved 17.07% on the accuracy of the natural language formulation analysis, which tested on search strategy. It shows the increment on recall and precision with 7% and 3%.

**Keywords:**semantic web; N-gram; ontology; statistical model.

Author Correspondence, e-mail: rabiahivi@ukm.edu.my

doi: http://dx.doi.org/10.4314/jfas.v9i4s.40

Journal of Fundamental and Applied Sciences is licensed under a Creative Commons Attribution-NonCommercial 4.0

## 1. INTRODUCTION

This paper presents the formation of natural language to the Resource Description Framework (RDF) triple entity using statistical language model and shows it is applied in search engine. The translation process of natural language is involved common natural language processing such as tokenization, stop word removal and lemmatization before its transform into RDF triples entity. The RDF format is very flexible where it can represent the triples entity in many ways in XML and it is able to fit with particular applications. It gives some standard ways of writing statements, therefore, it occurs in a document and produce the same effect in RDF terms. A single RDF assertion can be represented by the diagram called triple graph as shown in Fig. 1.

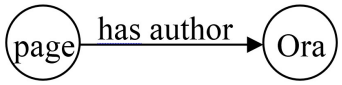


**Fig.1.** RDF assertion "The author of the page is Ora"

A triple entity pointed a statement in subject/predicate/object form, i.e. a statement links one object (subject) to another object (object) or a literal via a predicate.

Triple graph is used in semantic web to form a huge distributed knowledge based. Semantic web provides a common framework that allows data to be shared and reused across applications, which is able to enhance the search result by giving the meaning of a particular statement or query using RDF representation. Due to the ultimate usage of semantic web in helping users to locate, organize and process the information, there is no doubt that information is accessed or retrieved by human in a most comfortable language which is natural language. Therefore, a robust method is required to translate unstructured text i.e. natural language into structured form in RDF entity.

This paper proposed a statistical language model based on N-gram maximum likelihood estimation to translate natural language to RDF entity. The translation was applied to natural language queries from the Islamic Research Foundation founded by ZakirNaik. The triple entities are passed to SPARQL, which is then matched against the knowledge base for retrieval purposes. The experiment was tested on Islamic knowledge which contains the concepts from the English translation of Holy Quran.

Hence, this paper is aimed at predicting the concepts (subjects and objects) and predicate from natural language query to generate RDF triple entity and retrieve the relevant verses from the Holy Quran. The rest of the paper is organized as follows-related work on natural language query formulation will discuss in the following section, continue with the description of the proposed method in section 3. Section 4 will explained the implementation of the formation process. Experiment and result will show and discuss in section 5 and finally, the conclusion and future work in section 6.

## 2. NATURAL LANGUAGE QUERY FORMULATION

In recent years, there has been an increase in research interest on the semantic web and semantic search engines, so as to enable users to have easy access to structured data such as RDF triples representation in the knowledge base.This has motivated researchers to intensify efforts towards natural language query translation or formulation systems. In view of this, a considerable amount of literature has been published on semantically translate or formulate the natural language query into a structured query and retrieve relevant information or document from the knowledge base. However, due to the complex and ambiguous nature of natural language, different systems have been developed to cope with and address different natural language and technical issues. This has made the area subject to continued research consideration. The semantic search approaches have been adopted by different semantic query formulation systems.

Currently, there are many information available on the web, in databases, knowledge bases and other related document storage and manipulation mechanisms. Many important decisions that are made based on adequate support from information remains as a problem such as retrieval based on complex schemata [1]. This has led to a number of research problems that focus on the area of query formulation in order to enable easy access to such data [2]. The concept of the semantic web enables data to be stored in such a way that it is given well-defined meanings. In semantic web, data are presented in a structured form which requires the same structured query in order to process and retrieve the particular data. The natural language query formulation can be categorized into three approaches mainly manual, semi-automatic and automatic semantic query formulation systems. Most of those formulation

systems translated natural language queries into RDF triple format, and then to corresponding SPARQL to retrieve relevant information or documents from a knowledge base as recommended by the W3C group.

The manual formulation process involves user manually constructs a structured query language with a specific syntax according to the query language. Ontology editors such as Protégée and some query editors like Virtuoso SPARQL, Flint SPARQL Editor and Drupal SPARQL Query Builder enable users to input a formal query language and retrieve relevant documents from the knowledge base. The process of manual construction of the structured query is complex because it requires users familiar with the complex syntax of the query language. To improve the manual formulation process, semi-automatic natural language query formulation was proposed [3].

The second approach of formulation is semi-automatic where user adaption is required in the query formulation process. The user is interactively assisted by the system in order to translate the query. The semi-automatic natural language query formulation approach comprises of a combination of the automatic and manual approach where computers and humans collaborate to translate a formal query. Various approaches have been introduced for semi-automatic formulation such as template-based [4-5] and menu-based system [6-[7]. In this approach, users do not need to be familiar with the complex syntax of a structured query language before they can pose their query.

The final approach of formulation is an automatic translation process, which is adopted in this research. Several researches and methods have been proposed for an automatic translation process where the best method is selected as a benchmark in our experiment. This automatic process of natural language query formulation allows the system to translate a natural language query to a structured query without user intervention. Aqualog [8] is an example ofautomatic natural language translation which is applied on a factual question answering system i.e. what, where and when. The triple was generated using the Relation Similarity Service (RSS) approach. Aqualog then improved in 2011 with a system called Power Aqua [9]. Power Aqua was developed to be domain independent, where user queries do not have to target specific domains. However, one of the drawbacks of Power Aqua is that the system is limited to single sentence query. Then, the research of natural language translation is

continued using supervised machine learning technique [10], integer linear programming [11] and heuristic rules matching [12]. However, it will engage the user in resolving the ambiguity problem and produce several triples without rank it in order of precisely to the meaning of the query.

This paper discusses the statistical language model in automatic natural language translation by adopting WordNet to resolve ambiguous and Levenshtein-Algorithm to rank the triples by matching the string with the user query. The process of ranking is considered as reverse engineering of string matching approach. The detailed description will be explained in section4.

## 3. STATISTICAL LANGUAGE MODEL

The statistical language model was introduced to estimate the probability distribution of natural language as accurately as possible. Having a way to estimate the probability distribution of phrases, sentences and documents is useful in natural language processing applications such as information retrieval in the query likelihood model [13]. Commonly, the n-gram language model is used for this purpose, otherwise, it known as the bag of words model.

### 3.1. N-Gram Model

In the field of computational linguistics and probability, a n- gram is a contiguous sequence of n items from a given text. The n-grams typically are collected from a text corpus. It is a type of probabilistic language model for predicting the next item in such a sequence in the form of a (n-1) ordered by Markov model. In an n-gram model, the probability $P(w_1,\ldots,w_m)$ of observing the sentence $w_1,\ldots,w_m$ is approximated as:

$$P(w_1,\ldots,w_m)=\prod_{i=1}^{m}P(w_i \mid w_1,\ldots,w_{i-1}) \approx \prod_{i=1}^{m}P(w_i \mid w_{i-(n-1)},\ldots,w_{i-1}) \quad (1)$$

It is assumed that the probability of observing the $i^{th}$ word $w_i$ in the context history of the preceding i-1 words can be approximated by the probability of observing it in the shortened contextual history of the preceding n-1 words.

The conditional probability can be calculated from n-gram model frequency counts:

$$P(w_i \mid w_{i-(n-1)},\ldots,w_{i-1})=\frac{\text{count}(w_{i-(n-1)},\ldots,w_{i-1},w_i)}{\text{count}(w_{i-(n-1)},\ldots,w_{i-1})} \quad (2)$$

The word bigram and trigram language models denote n-gram model language models with n = 2 and n = 3 respectively [14-15].

### 3.2. Levenshtein-Algorithm

The Levenshtein-algorithm is a metric for measuring the amount of difference between two sequences ie. a distance. Mathematically, the levenshtein distance between two strings a and b is given by $\mathbf{lev}_{a,b}(|a|,|b|)$ where

$$\mathrm{lev}_{a,b}(i,j) = \begin{cases} \max(i,j) & \text{if } \min(i,j) = 0, \\ \min \begin{cases} \mathrm{lev}_{a,b}(i-1,j) + 1 \\ \mathrm{lev}_{a,b}(i,j-1) + 1 \\ \mathrm{lev}_{a,b}(i-1,j-1) + 1_{(a_i \neq b_j)} \end{cases} & \text{otherwise.} \end{cases} \quad (3)$$

where $1_{(a_i \neq b_j)}$ is the indicator function equal to 0 when $a_i = b_j$ and equal to 1 otherwise. The first element in the minimum corresponds to deletion (from a to b), the second to insertion and the third to match or mismatch, depending on whether the respective symbols are the same.

## 4. RDF ENTITY FORMATION

This section discusses the formation process of natural language queries to RDF triple entity representation in "subject/predicate/object". The formation process requires an input in text, and semantically relates the identified concepts via predicates to retrieve the knowledge base that contains of Islamic knowledge of the English translation of the holy Quran. To transform natural language queries into triple entity representation in RDF, the system automatically identifies concepts from the queries based on concept matching and attempts to automatically link them via predicates using the query words. If concepts are identified in the queries, those concepts are removed from the queries in order to use the remaining query tokens for further processing of detecting predicates that will link the identified concepts. The remaining query tokens, after concepts are identified and removed from them are used to automatically detect possible relationships between the identified concepts.

The formation has limitless of the natural language query either phrases, single sentence or paragraph of the query. Every query is parsed through the common natural language processing such as tokenization, stop word removal and lemmatization before the formulation. Then, the statistical language model is implemented based on N-gram maximum likelihood

estimation to automatically identify concepts and detect the predicates to generate RDF triples. Fig. 2 shows the framework of RDF entity formation, which involves several main processes such as concept identification, predicate detection, RDF generation and triples ranking.

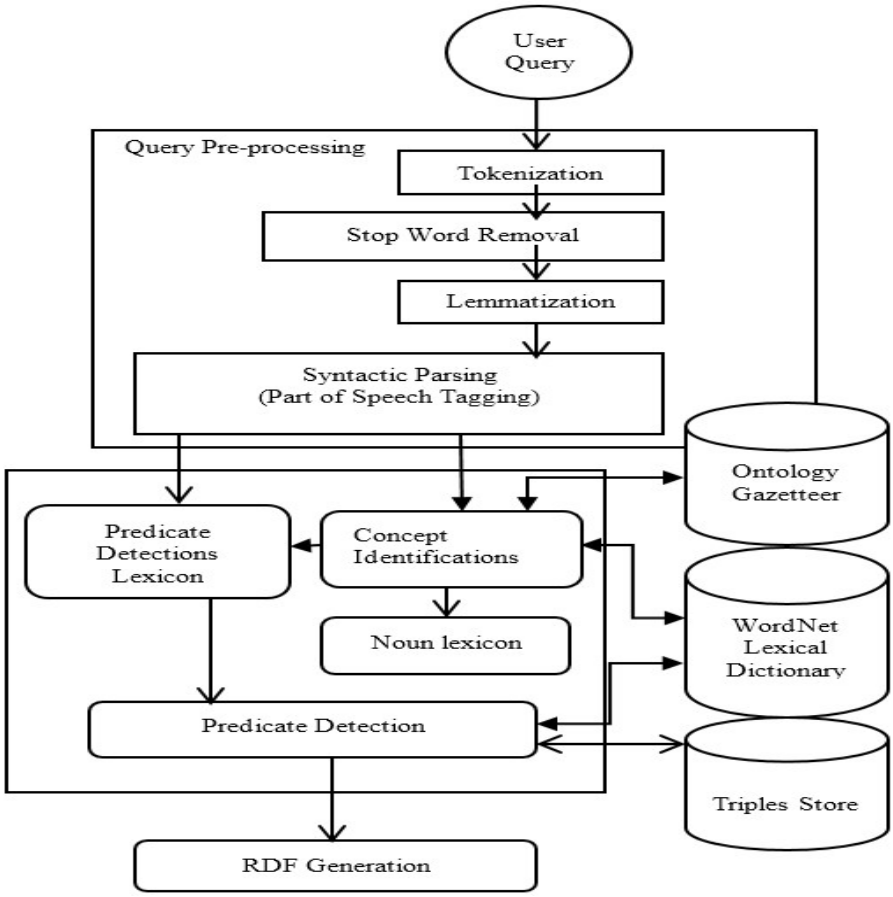


**Fig.2.** A framework of RDF entity formations

The Levenshtein string distance algorithm is executed to rank if more than one RDF triples are generated. It is a reverse engineering approach to get the most appropriate triple representation of the given natural language queries.

**4.1. Concept Identification**

The first process in the natural language query translation process is concept identification. Concept identification extracts any of the query tokens that are nouns as potential concepts. These potential concepts are then matched against ontology concepts in the list of Holy Quran ontology concepts called as gazetteer list. The gazetteer list contains 300 noun concepts from the Leeds University Quran ontology collected from the knowledge base. If any of the potential concepts match a concept on the gazetteer list, these potential concepts are identified as concepts and added to the concept lexicon. For example, a user may use the word

Muhammad in their query where Muhammad exists in the gazetteer list and thus it is identified and extracted from the query as part of the concept lexicon. Fig. 3 shows the work flow for the concept identification process. The identified concepts in a query, will then be used for further process of RDF triple translation by linking the concepts with explicit relationships or predicates.

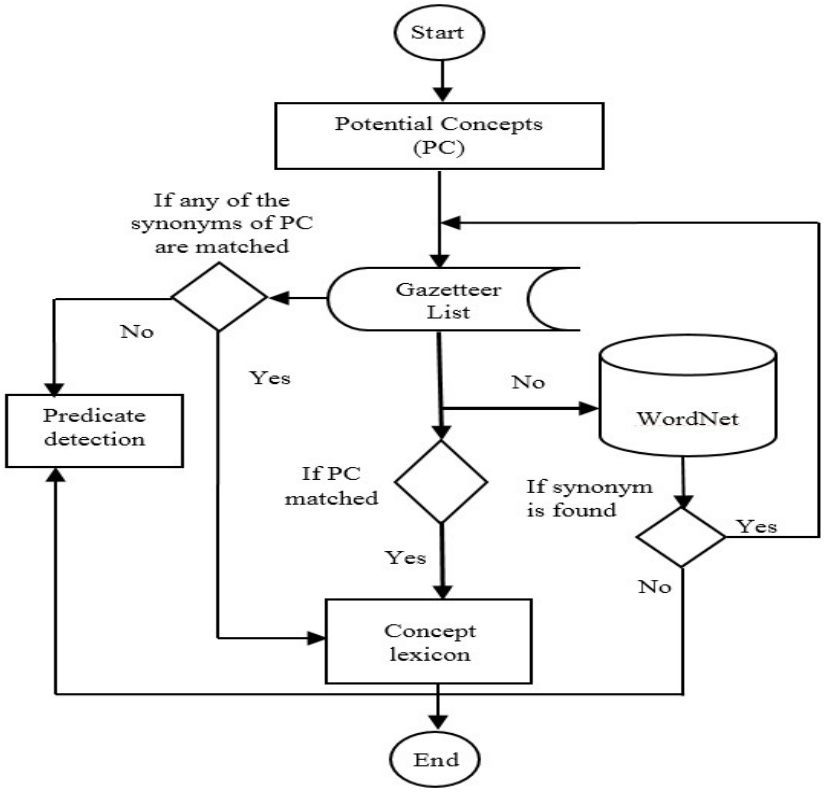


**Fig.3.** A flowchart of concept identification

### 4.2. Predicate Detection

Having identified concepts in the user's query tokens, it is important to link the concepts via predicates to create the relationships. Predicate detection involves scanning over the queries and detects possible relationships between the identified concepts. The detected predicates are used to link the identified concepts in order to form a RDF triples representation of the natural language query. Sometimes, an identified concept may also be linked with any other ontology concept in the knowledge base, even if such a concept is not identified from the query tokens. For example, (concept, predicate,?) or (?, predicate, concept) may possibly be translated where the missing question mark indicates the concept the user is looking for and such a concept is returned as an answer by the inference engine. For example, in "Who is the mother of Jesus?" the system will identify "Jesus" as a concept and then attempt to use the remaining

query tokens to detect the predicate that may link Jesus with any other concept in the knowledge base.

Predicates may be represented in the form of a word/phrase/sentence that links concepts to form a RDF triple representation (subject, predicate, object), where both subject and object/concept are ontology concepts with a word/phrase or sentence that relate the concepts/object. For example Quran, isAWordOf, God is a triple representation where Quran and God are ontology concepts and isAWordOf relates them to form a RDF triple representation of the query.

The statistical language model based on using N-gram maximum likelihood estimation is employed in predicates detection and the WordNet lexical dictionary for disambiguation. The proposed method is used to calculate the probability and estimate the maximum likelihood of possible predicates between the identified concepts. For example, computing the maximum likelihood for the sequence of words "is the father of":

$$P(W_1^n) = P(is)\,P(the/is)\,P(father/the)\,P(of/father)$$

Here, first probability is being the first unigram is computed $P(is)$.Then, the word is used to predict the next word by computing the probability that the next word will be the, $P(the/is)$. In this case, when the word the is predicted to be the next word after is, the next bi-gram is computed to predict the word after the current word the by computing the probability that the next word after the is Father $P(father/the)$. Bi-grams are computed until the nth-token.

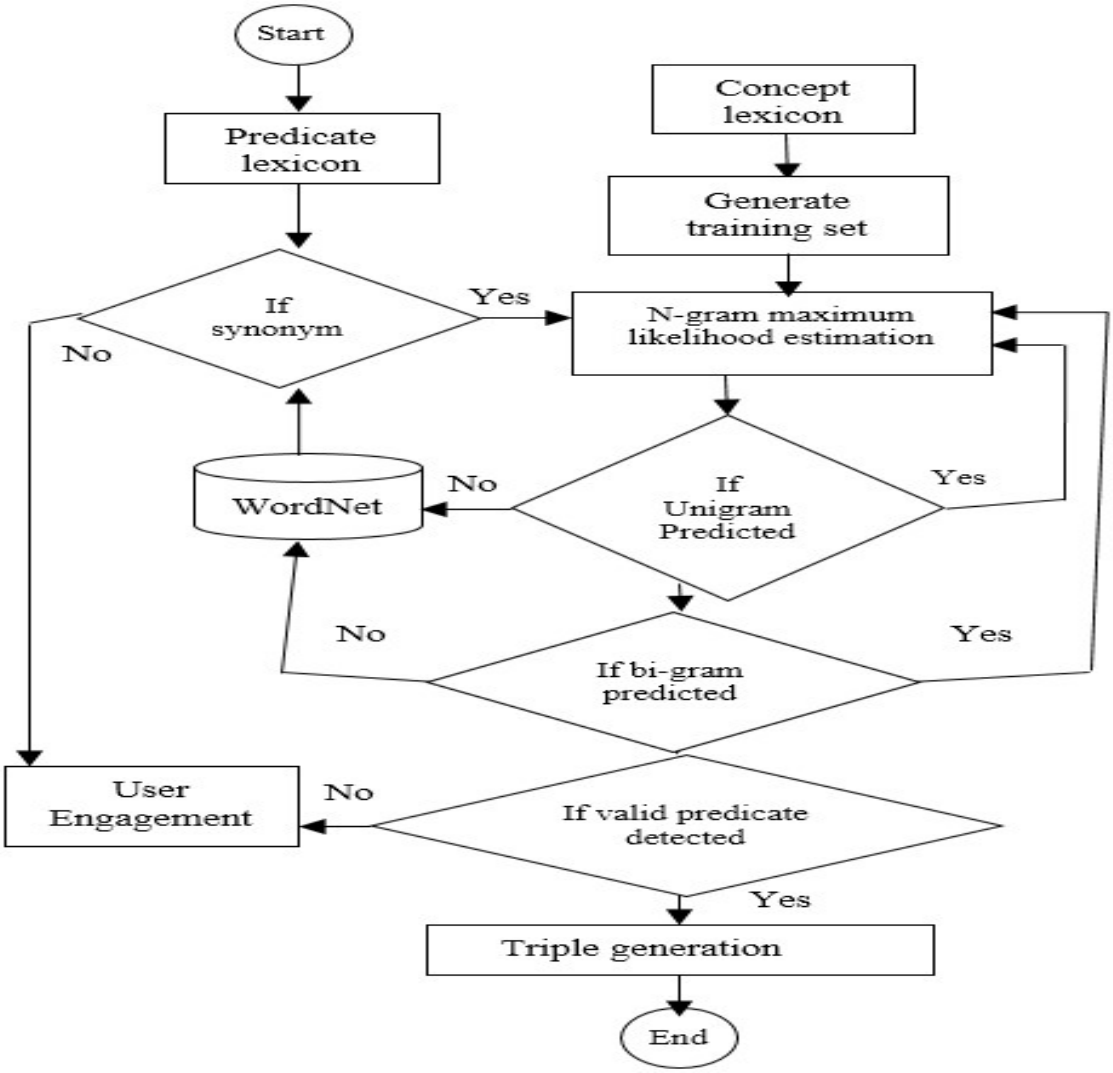


**Fig.4.** A flowchart of predicate detection

In this work, we modified the process of estimating the maximum likelihood for a phrase or sentence probability. Fig. 4 shows the workflow for the predicate detection process. Instead of multiplying or adding the series of estimated bi-gram probabilities, the process of predicting a phrase or sentence as a possible predicate between the identified concepts is performed by estimating bi-grams until a valid predicate is detected. A valid predicate here means any predicate that was used to link the identified concept with any concept in the knowledge base. The approach allows the system to learn from a training set and estimate the maximum likelihood of possible predicates between the identified concepts. The machine learning concept originates from the field of artificial intelligence, where it is mainly used for the creation and study of intelligent systems that learn from data.

The approach for predicate detection employs a supervised statistical machine learning approach to detect possible predicates between identified concepts by learning from the training set. The training data used contains all the triples in the knowledge base, which are automatically generated every time a query is posed by the user. That is to say, for every given query, the system automatically generates a training set based on the concepts that are identified in the query. For example, Prophet, God, Earth, Islam was automatically identified as concepts.Therefore, the training set for these identified concepts will be all the triples in the triple store that have predicates relating the identified concepts. Whereas, other concepts are automatically ignored. The predicate for the triples that are pulled are used as a training set for detecting possible predicates for the identified concepts. For every identified concept therefore, in order to detect possible predicates between that identified concept and any other concept in the knowledge base, a training set is automatically generated by the system.

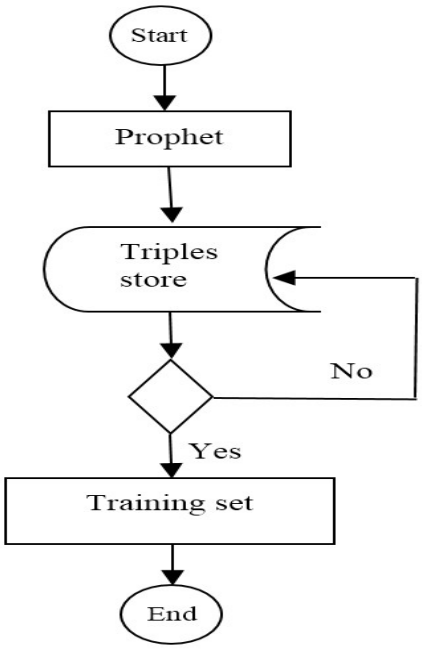


**Fig.5.** A flowchart of predicate detection

Fig. 5 shows how the system automatically generates a training set for an identified concept Prophet. The system automatically attempts to match the identified concept Prophet against the triple store. A triple store is a storage container for all the triples, subject, predicate, and object in the knowledge base. If Prophet matches the subject or object of any triple such a triple is automatically generated and all the predicates of the triples are added to the training set. The automatically generated training set is used for detecting possible predicates between Prophet and any other concept, either among the identified concepts or in the triple store.

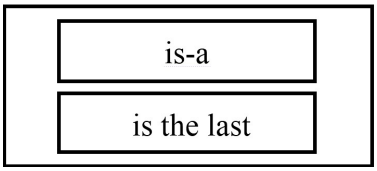


**Fig.6.** Sample of training set for 'Prophet' concept

Fig. 6 shows an example of the automatic generated training set that will be used for detecting possible predicates between Prophet and any other concept. The system is able to automatically generate the training set because Prophet matched objects of the triples Isah, is-a, Prophet and Muhammad, is The Last, Prophet from the triple store. The system generates different training sets for automatic predicate detection for each of the identified concepts in the concept lexicon.

Given a training set containing predicates from generated triples, the system uses the predicates to learn and identified concepts between predicates using the concept of N-gram language modeling. An N-gram is a language model for probabilistic automata for generating phrases or sentences based on predicting words given the previous word. For example, given the word I, the probability that the next word will likely be am. The N-gram language modelling give the probabilistic value of word prediction given by the previous word by assigning a conditional probability to the next possible words, which can be used to assign joint probability to an entire phrase or sentence.An N-gram is used to denote an N-token sequence of words. N-gram models are quite a powerful language modelling technique, based on the assumption that the current word depends on the preceding word, N-1. An N-gram is used as a statistical language modelling approach with the main aim of computing the probability of a word w given some history h, i.e. P(w|h). The most intuitive and simplest way of estimating N-gram probability is referred to as an N-gram maximum likelihood estimation. The N-gram maximum likelihood estimation estimates N-gram probabilities by counting relative frequencies in a training set and normalize the count to the interval of [0, 1]. The training set is used to estimate the maximum likelihood of a word, given the previous word. The general formula for N-gram maximum likelihood estimation is computed by taking the probability of the sequence of words given the previous word.

$$P(w_1^n) = P(w_1)\,P(w_2/w_1)\,P(w_3/w_2)\,P(w_2/w_1) \cdots\cdots P(w_n/w_1^{n-1}) \quad (4)$$

Equation (4) shows the general case for N-gram maximum likelihood estimation where $P(w_1^n)$ represents the probability of a sequence of words occurring together, either phrase or sentence

based on predicting a word given the previous word. $P(W_1)$is the unigram that will be used to estimate the first bi-gram $P(W_2/W_1)$, which estimates the maximum likelihood of the starting word $P(W_1)$ being followed by the next word $P(W_2)$. The next bi-gram will be estimated by predicting the next word given $P(W_2)$ i.e. $P(W_3/W_2)$. Bi-grams are computed by predicting the next word given the currently estimated word until the $n^{th}$ i.e. $P(W_n/W_1^{n-1})$. The product of the count of estimated bi-grams is the maximum likelihood estimation of a sequence of words occurring together, i.e. a phrase or sentence $P(W_1^n)$. For example, computing the maximum likelihood for the sequence of words "is the father of":

$$P(W_1^n) = P(is)\,P(the/is)\,P(father/the)\,P(of/father)$$

Here, first probability is being the first unigram is computed $P(is)$.Then, the word is used to predict the next word by computing the probability that the next word will be the $P(the/is)$. In this case, when the word the is predicted to be the next word after is, the next bi-gram is computed to predict the word after the current word the by computing the probability that the next word after the is Father $P(father/the)$. Bi-grams are computed until the nth-token as seen in Equation (2).

$$P(W_n/W_{n-N+1}^{n-1}) = \frac{Count(W_{n-N+1}^{n-1}W_n)}{Count(W_{n-N+1}^{n-1})} \quad (5)$$

The formula in Equation (5) is used to compute the probability of a sequence of words by computing the sequence of the bi-gram and multiplying or adding the probabilities in order to obtain the best possible prediction of a phrase or sentence. That is to say, when the first bi-gram is computed, the next bi-gram will focus on computing the next word after the predicated word in the first bi-gram. The general formula for maximum likelihood estimation bi-gram probability is represented in Equation (6).

$$P(W_i/W_{i-1}) = \frac{count(W_{i-1}W_i)}{count(W_{i-1})} \quad (6)$$

Equation (3) shows the maximum likelihood estimation for b-gram probability. The probability $P()$ that a particular word token or word $W_n$ will precede a token or word $W_{n-1}$ is

the probability of their bi-gram. Here, the system takes the co-occurrence of two tokens $W_n$ and $W_{n-1}$ and divides by the probability of the proceeding token or word $W_{n-1}$. So, computing the maximum likelihood estimate of bi-gram probability $P(W_i/W_{i-1})$, i.e. the probability that the next word is $W_n$ given the previous word $W_{n-1}$ is computed by dividing the number of times $W_n$ and $W_{n-1}$ occur together by the number of times the word $W_{n-1}$ occurs in the training data. This will enable the computation to be normalized to the range between 0 and 1. The possible bi-gram with highest weight i.e. $W_n$ that proceeds $W_{n-1}$ is predicted to be the next possible word given the previous word. So, in order to obtain the maximum likelihood of a phrase or sentence, the counts of estimated bi-grams are multiplied to obtain the most likely phrase or sentence given a training set. The probability of the maximum likelihood estimation for a phrase or sentence could also be represented in the form of log probabilities by adding the counts of the estimated bi-grams instead of multiplying.

In this research, we modify the process of estimating the maximum likelihood for a phrase or sentence probability. Instead of multiplying or adding the series of estimated bi-gram probabilities, the process of predicting a phrase or sentence as a possible predicate between the identified concepts is performed by estimating bi-grams until a valid predicate is detected. A valid predicate here means any predicate that was used to link the concept “Prophet” with any concept in the knowledge base. For example, a valid predicate here is either “is-a” or “is the last” is automatically estimated. This is because in order to retrieve information from the protégée which serves as a knowledge base for retrieval, queries have to be semantically translated to match exactly any of the triples in the protégée. Until we obtain a valid triple, the system automatically computes for the possible bi-grams. The proposed approach does not try to obtain the phrase with the highest weight as the possible detected predicate. Even if the phrase or sentence with the highest weight is selected as the possible predicate as long as it cannot match any triple in the knowledge base, the system may not return an answer for the given query. The proposed approach automatically continues computing for bi-gram probability, by predicting the next word, given the history and the predicted word is then used to compute the next word until a valid predicate is generated.

For example, if a user posed a natural language query and the concepts were automatically identified from the query, the proposed approach attempts to automatically detect predicates by using the concept lexicon and predicate lexicon. First, the system reads the concept lexicon and automatically generates a training set as described earlier in Fig.8. For purposes of demonstration, we will only show examples for detecting possible predicates between the identified concept Prophet and any other concept either among the identified concepts or any other concepts in the knowledge base. Fig. 9 therefore shows examples of the automatically generated training set that will be used for detecting possible predicates between Prophet and any other concepts.

In order to have a consistent probabilistic model, the system automatically affixes a unique start (<s>) and end (</s>) symbol to every predicate in the training set to indicate the start and end of the predicate. The proposed approach uses the additional symbol <s> at the beginning and end of predicates to automatically identify the start of the predicate, which will be used for unigram estimation and </s> indicating the end of a predicate that assists in identifying when a valid predicate has been detected. Fig. 7 shows how the training set is automatically processed by affixing start and end symbols in order to facilitate further processing.

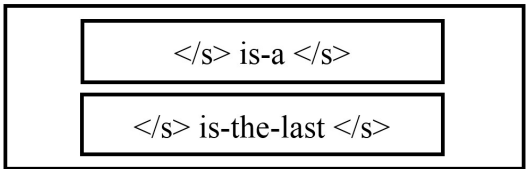


**Fig.7.** Training set by affixing the </s> symbol

### 4.3. RDF Generation

The RDF triple entity representation of the natural language queries provides a semantically structured query format. The triple entity formation process is the linking of identifying concepts with the predicate detected. The system automatically links the detected predicate with the concepts that were linked via such a detected predicate. For example, the system automatically forms a triple entity by linking the relationship with concepts that are related via the predicate such as is-the-last and is-a were detected. A concept in the knowledge base is related to prophet via the predicate is-the-last and is-a as seen in Fig. 8 and Fig. 9 respectively.

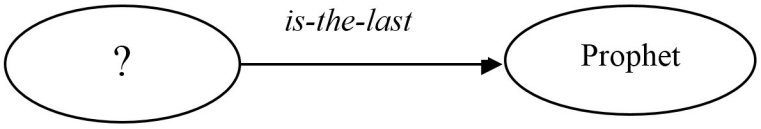


**Fig.8.** Diagram of RDF entity relation </s> is-the-last </s>

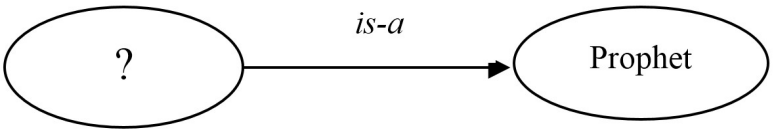


**Fig.9.** Diagram of RDF entity relation </s> is-a </s>

These RDF triple entities are parsed to the retrieval module to retrieve an answer from the knowledge base. Several possible triples may be formatted when the system is able to detect potential predicates between the identified concepts as seen in Fig.7 and Fig. 8. This research proposed a mechanism to rank and select the triple that is closer to the user query. In order to be more precise and return a closer triple entity representation, Leveinshtien string matching reverse engineering is implemented for RDF triple entity ranking.

**4.4. Triples Ranking**

Triple ranking is required when more than one possible RDF triples entity representation of a query are returned by the triple generation process. When more RDF triples entity of a query is returned, it makes the process of retrieval difficult. Therefore, we need to rank the returned RDF triples in order to get most appropriate RDF triple representation for a given natural language query. Levenshtein string similarity algorithms propose to compute the distance between two given strings.

In order to get the most suitable RDF triple representation of the query, the Levenshtien string matching algorithm matches the generated RDF triple entity against the triples in triple store and get the score of string similarity. The generated RDF triple entity with few strings is likely to come closer in similarity because there is less insertion and substitution and it be more appropriate RDF triple entity representation of the query.

The operations are comprised of insertions, deletions or substitutions of a single character. Levenshtien computation is normalized and assigned a score in the range 0 and 1. For example, using the Levenshtein distance between "kitten" and "sitting" is 3, i.e. the number of operations comprising the substitution and insertion needed to convert kitten to seating is 3. Levenshtien string matching algorithm matches the generated triple against the triples in triple stored to get triples with the highest score on top, and then use the ranked triple to perform reverse engineering ranking by matching the ranked triples against the original natural

language queries. The reason for reverse engineering is that based on experiments conducted, most of the ranked triples by matching generated triples against the triples in the triple store are returned triples that are not the most appropriate triple representation of the queries with highest similarity because Levenshtien is based on the string distance which is based on inserting and substitutions of strings. So, a generated triple with few strings is likely to come closer in similarity to another triple in the triple store because there is less insertion and substitution, even though it may not be the most appropriate triple representation of the query. However, this initial ranking reduces the number of triples when a few dozen triples are returned. From the initial ranking, the top five triples are returned and a reverse engineering approach is implemented. Reverse engineering was therefore implemented by using the Levenshtien string matching algorithm to match the initial top five ranked triples against the original queries, which resulted in obtaining the most appropriate triple representation of the queries being ranked with the highest score.

For example, the triples as shown in Fig. 7 and Fig. 8, the triple ranking process starts by comparing the generated triple with those in the triple store using a Levenshtien string matching algorithm training set. The ranking returned,?,is-a, Prophet has a higher weight than ?,is-the-last, Prophet. By applying a reverse engineering of Leveinshtien string matching, the minimum distance in transforming any of the ranked triples is considered close to the original query. This approach gave better results in terms of obtaining the triple representation most appropriate to the triple representation with the highest score. For example, after applying the reverse engineering approach,?, is-the-last, Prophet has a higher score and thus is accepted by the system as the semantically translated triple. After ranking the triples, the top ranked triple is used is parsed to the retrieval module for retrieval of the relevant Holy Quran verse.

## 5. RESULTS AND DISCUSSION

Leeds University. It contains of 300 nouns, i.e. noun concepts and 1475 verbs. Number of predicates used for the experiment contained 350 relationships. A total of 82 natural language queries obtained from the Islamic Research Foundation website were used for the question answering experiment. The best existing approach i.e. heuristic rules matching used in FREyA, of natural language query formulation was chosen as our benchmark of the performance. Table 1 shows the results for the evaluation of the correctness of the RDF triples translated queries. The percentage of the queries that were correctly translated is annotated as “Correct”, the percentage of the queries that were not translated correctly due to lack of

correspond knowledge in the knowledge base is notified as “No Answer” and “Failed” is the percentage of queries that failed to be translated even though there was a corresponding knowledge in the knowledge base for semantically formulating the queries were existed.

**Table 1.**Correctness of the natural language query to RDF triples translation

| | **No. of Queries** | **Percentage (%)** |
|---|---|---|
| No. of queries | 82 | |
| Statistical Language Model | | |
| Correct | 74 | 90.24 |
| No Answer | 6 | 7.32 |
| Failed | 2 | 2.44 |
| Heuristic Rules Matching | | |
| Correct | 60 | 73.17 |
| No Answer | 6 | 7.32 |
| Failed | 16 | 19.51 |

The evaluations of the retrieval effectiveness retrieved were measured on Holy Quran Verses. The effectiveness of the retrieval was measured based on precision and recall. Precision and recall are the common methods to measure the effectiveness of the search system. Recall measures how many of the relevant documents were retrieved, while precision measures how many of the retrieved documents were relevant. Table 2 depicts the result of the precision and recall between the proposed method and heuristic rules matching in FREyA.

**Table 2.**Performance of precision and recall

| **Characteristic Statistics** | **Precision** | **Recall** |
|---|---|---|
| Statistical Language Model | 0.61 | 0.69 |
| Heuristic Rules Matching | 0.58 | 0.62 |

The results show that the proposed method had a precision of 0.61 and recall of 0.69, while the in FREyA had a precision of 0.58 and recall 0.62. It shows the increment on recall and precision with 7% and 3% respectively. Therefore, the proposed method isoutperformed than FREyA in terms of precision and recall.

## 6. CONCLUSION

The main idea of this research was to examine the process of the natural language to RDF triples translation of natural language queries in order to retrieve Holy Quran Verses. Holy Quran ontology was used as a test-bed to translate natural language queries. The translation of natural language queries to structured queries involves solving ambiguous natural language and transforming it into formal structured query language. Formal structured query language enables the retrieval of structured data in RDF triple format. Simplifying access to structured data needs a system that semantically accepts natural language queries and semantically transforms such queries to structured formal queries for retrieval from the knowledge base. One of the biggest challenges of semantically transforming natural language query to structured queries is the ambiguity in natural language.

Our future work will include building a large Quran knowledge base that will support a variety of Quranic applications and facilitate the retrieval of Quranic knowledge. We also intend to incorporate the Hadith in the knowledge base. By doing so, a semantic search will be able to retrieve corresponding Quran verse as well as related Hadith.

## 7. ACKNOWLEDGEMENTS

Our thanks to Universiti Kebangsaan Malaysia (UKM) for funded this research under research grant code GGPM-2015-003.

## 8. REFERENCES

[1] Gobet F, Lane P C, Lloyd-Kelly M. Chunks, schemata, and retrieval structures: Past and current computational models. Frontiers in psychology. 2015, 6:1-4

[2] terHofstede A H, Proper H A, van der Weide T P. Computer supported query formulation in an evolving context. Australian Computer Science Communications, 1995, 17:188-202

[3] Damljanovic D, Agatonovic M, Cunningham H. FREyA: An interactive way of querying linked data using natural language. In 8th European Semantic Web Conference Workshops 2011, pp. 125-138

[4] Kiryakov A, Popov B, Ognyanoff D, Manov D, Kirilov A, Goranov M. Semantic annotation, indexing, and retrieval. In 2nd International Semantic Web Conference, 2003, pp. 484-499

[5] Stratica N, Kosseim L, Desai B C. Using semantic templates for a natural language interface to the CINDI virtual library. Data and Knowledge Engineering, 2005, 55(1):4-19

[6] Zenz G, Zhou X, Minack E, Siberski W, Nejdl W. From keywords to semantic queries-Incremental query construction on the semantic web. Web Semantics: Science, Services and Agents on the World Wide Web, 2009, 7(3):166-176

[7] Aksac A, Ozturk O, Dogdu E. A novel semantic web browser for user centric information retrieval: PERSON. Expert Systems with Applications. 2012, 39(15):12001-12013

[8] Lopez V, Uren V, Motta E, Pasin M. AquaLog: An ontology-driven question answering system for organizational semantic intranets. Web Semantics: Science, Services and Agents on the World Wide Web, 2007, 5(2):72-105

[9] Lopez V, Fernández M, Motta E, Stieler N. Poweraqua: Supporting users in querying and exploring the semantic web. Semantic Web, 2012, 3(3):249-265

[10] Lehmann J, Bühmann L. Autosparql: Let users query your knowledge base. In G. Antoniou et al. (Eds.), Thesemantic web: Research and applications.Berlin: Springer, 2011, pp. 63-79

[11] Lampouras G, Androutsopoulos I. Using integer linear programming in concept-to-text generation to produce more compact texts. In51st Annual Meeting of the Association for Computational Linguistics, 2013, pp. 561-566

[12] Yahya M, Berberich K, Elbassuoni S, Ramanath M, Tresp V, Weikum G. Natural language questions for the web of data. In Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, 2012, pp. 379-390

[13] Damljanovic D, Agatonovic M, Cunningham H. Natural language interfaces to ontologies: Combining syntactic analysis and ontology-based lookup through the user interaction. In L. Aroyo et al. (Eds.), The semantic web: Research and applications. Berlin: Springer, 2010, pp. 106-120

[14] Chen Z, Zhu Q. Query construction for user-guided knowledge discovery in databases. Information Sciences, 1998, 109(1-4):49-64

[15] Manning C. D., Schütze H. Foundations of statistical natural language processing. Cambridge: MIT Press, 1999

**How to cite this article**:
Kadir R A, Yauri R A. Resource description framework triples entity formations using statistical language model. J. Fundam. Appl. Sci., 2017, *9(4S), 710-729.*