Journal of Theoretical and Applied Information Technology
31st October 2015. Vol.80. No.3

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

LINGUISTIC RULE-BASED TRANSLATION OF NATURAL LANGUAGE QUESTION INTO SPARQL QUERY FOR EFFECTIVE SEMANTIC QUESTION ANSWERING

NURFADHLINA MOHD SHAREF[1], SHAHRUL AZMAN NOAH[2], MASRAH AZRIFAH AZMI MURAD[3]

[1,2]Intelligent Systems Research Group, Faculty of Computer Science and Information Technology, University of Putra Malaysia, Malaysia

[1]Knowledge Technology Group, Centre of Artificial Intelligence, Faculty of Technology and Information Science, National University of Malaysia

E-mail: nurfadhlina@upm.edu.my, samn@ftsm.ukm.my, masrah@upm.edu.my

ABSTRACT

Semantic question answering (SQA) demands different processing compared to the common information retrieval method because the semantic knowledge base is stored in the triples form. However, manipulating the knowledge requires understanding of its structure and proficiency in semantic query language such as SPARQL. Natural language interface (NLI) alleviates this by allowing user to input question in their human language. Then it produces an answer by translating the input into an equivalent SPARQL before it is executed to retrieve the answer. However, none of the existing research has presented a holistic computational model for the translation of NL question into an equivalent SPARQL for the semantic KB querying. Existing studies have focused mainly on the semantic disambiguation through consolidation where user interaction is initiated so that relevant concept can be chosen by the user to be inserted into the SPARQL. Besides, the linguistic understanding of the input has limited coverage where only one triple is constructed which loses many original expressions. There is a necessity to increase the linguistic understanding to involve multi-variables and multi-triples in the translated SPARQL so that accurate answer will be returned. Therefore, in this paper the linguistic challenge in NLI is addressed, specifically on the question complexity depth, processes that need to be performed to answer the question and gaps in existing study. A linguistic-rule-based translation model for natural language question is introduced that utilizes a set of observational variables to extract the information in the question; (i) checking if the focus is equals to subject, (ii) number of subjects, (iii) number of property, (iv) number of object, (v) checking if object is instance, (vi) checking if the question contains superlative expression, (vii) superlative orientation and (viii) checking if the question contains aggregates expression. The model is also aimed to reduce dependability on clarification dialogues. The results
show that the approach has managed to eliminate clarification dialogues and increase linguistic coverage of NLI.

Keywords: *Semantic Question Answering, Natural Language Interface, Natural Language, SPARQL, Semantic Search, Semantic Web*

1. INTRODUCTION

Semantic web (SW) technologies offer potential enhancement in current knowledge-driven question answering (QA) systems [1]. The linked data initiative for instance offers huge resources of knowledge waiting to be tapped and queried. However, querying the so-called semantic-web enabled knowledge base (KB) requires the user to understand the formal query language such as the SPARQL Protocol and RDF Query Language (SPARQL) as well as the KB schema. Therefore, it is valuable to replace the difficulty of constructing a formal query with natural language interface (NLI) that allows user to construct questions using natural language. NLI consists of the production of an intermediate logical representation from the input and mapping the input into a form which is consistent with the KB [2]. NLI can be divided into three types of input: keyword-based input, controlled natural language (CNL) and full NL.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

The keyword-based method [3] has limited ability to link the semantic expression of the input with appropriate operations in SPARQL. For example the question '*What is the longest river in Florida?*' where the expression '*longest*' should be translated as the descending operation ('DESC') in the SPARQL query that refers to some concept '*river*'. In this case the keyword-based method is unable to map such natural language expression to the appropriate SPARQL aggregate expression. The CNL method on the other hand when posed with the same questions will iteratively checks the input and suggest list of words according to the content of KB. This inherently limits the input flexibility which require substantial amount of user interactions. Therefore, full NL input is the most convenient querying method [4].

However, full NL input incurs many challenges such as the ability to process the ambiguity inherited by the linguistic complexity in the question and the mapping of NL question with the KB structure. This is because question does not necessarily matches the structure of the knowledge in the SW data besides various complexity level of the NL input. NL input complexity (Sharef & Noah, 2013) can be divided into three patterns namely *selection*, *arithmetic* and *composition* as shown in *Table 1*.

Table 1: Natural Language Question Complexities

Pattern	Example	Number of triples required	Remark
Selection	What is the capital city of Texas?	Single	Simplest
Arithmetic	What is the state with highest population?	Single/Multiple	Involves constraint operation
Composition	What is the capital city of the state with the highest population	Multiple	Involves constraint operation Consists of sub-questions

The *selection* pattern is being addressed by most of the existing NLI because it is the easiest form of computation. The *arithmetic* question processing is handled in ORAKEL [6] but demands lots of customization to be reused because the linguistic argument structures are mapped to the relations in the KB [7]. FREyA [8] attempted *arithmetic* question processing by adopting clarification dialogues where user disambiguates the terms in their question according to the KB structure but to the cost of confused user and may lead to wrong answer. The *compositional* question processing has been attempted only in PowerAqua [9] however was not tested formally. Besides this, the technique used in the NL input to SPARQL query translation is mostly based on single triple (constituted by <subject, property, object>) rewriting which constraints the expression rewriting. Most NL-to-SPARQL translation techniques mainly focus on a specific pattern and ignore the other patterns.

Therefore, it is the interest of this research work to propose a technique which can be used for translating NL questions involving all the aforementioned three patterns. In this paper we introduce a NL-to-SPARQL translation model based on observational variables for deeper semantic question understanding consisting of:

(i) the *focus* of the question [10],
(ii) the number and types of concepts used, and
(iii) the constraint types.

For example, the NL question '*What is the capital city of the state with the highest population*?', the proposed technique called MYAutoSPARQL will identify the '*capital city*' as the *focus*, with '*state*' as the *subject*, capitalOf as the object property and '*population*' as *datatype* property. The '*highest population*' will be identified as *constraint*. The MYAutoSPARQL translation model is tested on various NL input complexities and compared against FREyA [11]. FREyA is an NLI system that utilizes clarification dialogue so that the user can assist the system by disambiguating terms.

The paper is organised as follows. The first part of the paper introduces the NL to SPARQL translation while part two presents the existing approaches in SQA. The third section presents the challenges in the NL to SPARQL translation followed by the model. The next part provides the results from the evaluation of the translation model against a benchmark system. The final

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

section concludes the paper and presents future work.

2. EXISTING APPROACHES IN SEMANTIC QUESTION ANSWERING

Many research works on NL QA is based on information retrieval approach [12] which matches the content of the dataset in the corpus with the semantic-expanded keywords in the query. However, this does not guarantee that the answer is precise, as only matching document will be returned as results. Semantic web enabled KB querying through early NLI systems such as Querix [13], NLPReduce [14] and ORAKEL [7] allow limited NL question construction indicated by questions that begin with 'What', 'Which', 'How Many' and 'Does'. This approach clearly constraints the expressiveness of semantic questions.

In order to allow semantic concepts querying, SemSearch [15] implements a Google-like Question Interface. However, SemSearch requires a list of concepts (e.g., classes, instances) as an input and separated by colon. Although this method reduces ambiguity problem in processing the question, this indicates that the user of SemSearch needs to have some knowledge on the schema of the KB. Besides, more flexible and expressive question input should be provided.

Improved NL input flexibility is processed by AquaLog [2] and PowerAqua [9] by dynamically identifying around 14 different question categories so that the system can process the solution. The categories include basic NL input requiring an *affirmation/negation* or a *description* as an answer; or the big set of queries constituted by a *wh-question*, where the relation is implicit or unknown or where no information about the type of the expected answer is provided. NL input that have two explicit relationships between terms such as containing *conjuction* or *auxiliary* verb, and a query can be a composition of two basic questions are also processed. In this case, the intermediate representation usually consists of two triples, one triple per relationship. The answer generation is driven by the triple categories and combination of the triples. However, identification of the NL input type is not always sufficient to find answers such as losing the main focus in the question.

FREyA [10] combines the syntactic parsing output of the question with heuristic rules in order to find the *focus* and the *answer type*. The *focus* is identified using syntactic parsing while the *answer type* is identified based on detection of the first ontology concept (FOC) in the question; whether a class or datatype property type. If the FOC refers to an *object property*, consolidation is performed with the domain or range of the classes of the *object property*, while if the FOC refers to an *instance*, consolidation is performed with a class of that *instance*. Besides the identification of question *focus* and *answer type*, the linguistic triples in the NL input are extracted [9,11]. The candidate triples are disambiguated in order to form semantic triples. Several approaches were adopted for this purpose such as utilizing string similarity matching, WordNet, lexicon and clarification dialogue. While the first three aims for automation, the fourth is executed to let user to choose the closest ontology concept that represents his intention. Users are involved to disambiguate their intention (i.e., when the relation is implicit or unknown or unclear) guided by confidence score (frequency of the suggestion being chosen). However, clarification dialogue is only efficient if the users have some background knowledge on the ontology scheme. Otherwise, it may be confusing to be used. Furthermore, correct result may not be reached if the interaction with the dialogue has lead towards wrong navigation and could not be answered by the reasoning engine.

Besides the problem caused by the clarification dialogues, the extent of linguistic coverage in FREyA, AquaLog and PowerAqua could be improved for example on the composite semantic question processing that requires multi-triples and constraint operations. To the best of our knowledge ORAKEL (Cimiano et al., 2007) is the only NLI that supports questions that contain compositional semantic construction so it is able to handle questions involving *quantification*, *conjunction* and *negation*. This is because of its support for questioning knowledge represented in OWL besides a means for accessing F-Logic. However, ORAKEL usage requires mandatory customisation because the linguistic arguments are mapped to the domain in development which may be overload to the user. Instead, an ideal NLI should be modular with respect to the target KB, and there is zero or minimal cost to switch from one KB to another.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

More recent research works for NLI have emphasized on the deep analysis of the queries and the techniques involved to translate the NL question into a SPARQL compliance [11,16–18]. Most approaches rely on the linguistic triple identification (*subject*, *predicate* and *object*) besides identifying answer entity type as class, *data property*, *object property*, annotation, axiom, *instance* and entity location in the ontology. The linguistic triple is then translated as a single triple in the SPARQL syntax before it is reasoned by the NLI engine to generate the correct answer. However, relying on linguistic triples identification is insufficient because of the complexity in the NL question patterns. This is due to the ambiguity issue to match the detected linguistic triples with the matching ontology triples and the composite question which contains information indicating constraints such as the FILTER expression, *arithmetic* operation (e.g, COUNT, SUM, etc.), *comparative* process (e.g, sort descending and ascending), and *negation* (e.g, outside, not, besides, etc.).

[19] proposed composite questions to be decomposed into atomic (simpler) ones so that they can be answered directly using existing QA capabilities. The final answer is generated by combining results from the atomic questions computation corresponding to the question type. Three facets are introduced; *answer format*, *answer theme*, and *question qualifier*, to characterize the questions and prescribing system functionality accordingly. The answer format has four possible values: single, multiple, descriptive, and yes/no. The answer theme is the class of the object sought by the question, such as PERSON, LOCATION, and DATE. The question qualifier indicates the semantic or pragmatic nature of a question and requires corresponding operations such as specification, superlative, ordering, definition, reason, method, comparison, negation, report and analysis. However, this paper is not tailored to semantic QA although the composite question characterization theory is very beneficial in classifying questions according to their complexity.

A variant of NL input complexity classification [20] emphasizes the number of triples and variables that are required to answer the question. There are four fact-oriented classes proposed namely Type 1: Single variable and single triple, Type 2: Multiple variables and single triple, Type 3: Single variable and multiple triples and Type 4: Multiple variables and multiple triples. However, the scope of their work covers only query Type 1 and 3.

In this paper we present a method that can process questions of Type 1, Type 2, Type 3 and Type 4. Based on our preliminary work [5] on SPARQL construction for question answering, the methods that depend on the answer type solely are not sufficient in ensuring the coverage of the translated SPARQL from the question. This is because most of the translator system generates the SPARQL by utilizing the mapped identified predicate to the candidate properties from the ontology (typically the candidate properties are identified based on the loaded properties shared between the class of the subject and the class of the object). In fact, more than one property can be involved and this scenario is more prominent when composite queries such as those involving constraint expressions.

We also focus on the automation of disambiguation in our NL-to-SPARQL translation model by adding a set of observational variables. Our model enriches the existing question processing variables i.e., *question focus*, *answer type* and *linguistic triples*. We provide a deeper mining on the relationship among each of the variables, and do not restrict the linguistic triples to be in the standard <subject,predicate,object> form. Rather, we annotate the NL question according to possible ontology matching and form the linguistic triples and subtriples (when necessary). This means we process more than one class and property, which may also result to multi-triples. In the next section we describe the challenges in NL to SPARQL translation that can cover all four fact-oriented NL input types followed by a section on the translation model.

3. CHALLENGES IN TRANSLATING NATURAL LANGUAGE QUESTION INTO SPARQL QUERY

We focus on the fundamental challenges in translating an NL question into its equivalent SPARQL under the question complexity perspective and semantic ambiguity in input-ontology mapping. This paper addresses these challenges through the translation model presented in the next section.

3.1 Question Complexity

In reformulating the NL question into its equivalent SPARQL query, the question needs to be analysed to determine the type of processing

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

needed. The type of questions can be characterised by three facets [19] namely answer format, answer theme, and question qualifier (as shown in Figure 1). However, [19] method is more suited to information retrieval based QA and is not enriched by the semantic representation.

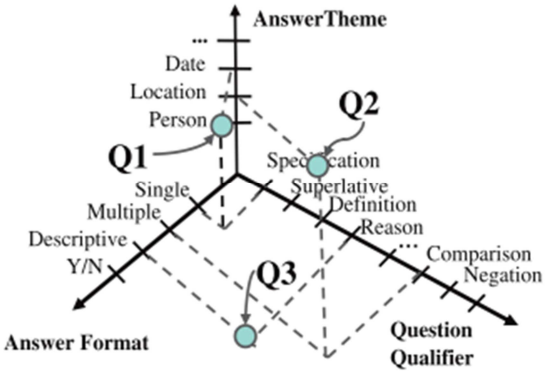

Figure 1: Conceptual Space with Three Facets for Question Types [19]

Figure 2 shows the perspective on question complexity consideration for semantic QA or NLI which covers four aspects namely the *focus* of the question, the *depth* of the question, the *answer type* and the *operation* needed in answering the question. The *focus* of a question can either be a class, a property, or an instance in the ontology. Identifying the *focus* is important to return a precise answer and also to manipulate the content of the ontology. The *depth* of the question determines the number of triples needed in answering the question.

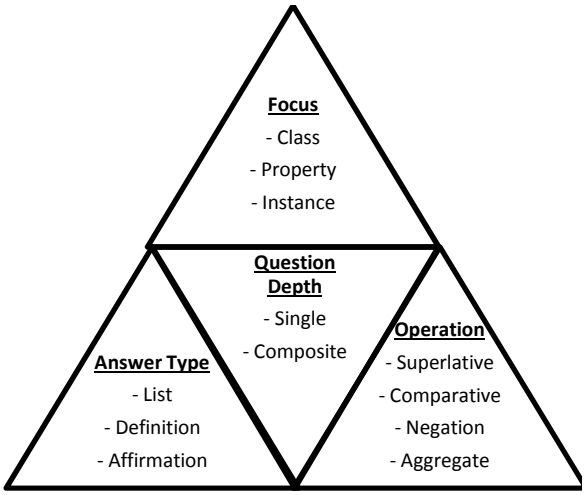

Figure 2: Perspective on Question Complexity

Despite various question categorization methods in a semantic QA the triples determines whether the information requirement expressed in the question has been captured properly. The *answer type* determines the final answer to the question, such as a list of matching pattern based on the constructed SPARQL, a definition or affirmative answer. The *operations* needed to obtain the answer can generally be identified by detecting a vocabulary of indicative expressions; however the accuracy of the answer depends on the correct positioning of the operator amongst the triple.

3.1.1 Focus

The *focus* of the question indicates the most important variable that drives the translation of the input into triples to generate the final answer. The *focus* used in this paper is also known as the answer type in previous existing similar works [11]. The *focus* is identified based on the first encountered ontology concept. However, linguistic triple and *focus* are only sufficient for the computation of direct questions; which does not involve any constraint expressions.

There are three types of *focus* that relates to the kind of concepts needed as the answer namely class, property and instance. It is also identified based on the nearest encountered ontology concept to the question header. For example, in 'What is the capital of Texas?', the *focus* is 'capital'. Once the linguistic triples are identified (in this case <?, capital, texas>, the connected properties between the *focus* and the linguistic property are loaded, resulting to 'capitalOf'. Then, the triple can be constructed to generate the answer to the question.

3.1.2 Question depth

The depth of a question can be categorised according to the number of triples needed to produce the answer. The selection-typed question indicates that only one triple or a maximum of one variable is used in one triple needed in the translation. A more challenging scenario exists when dealing with composite questions that involve multi-triple and multi-variables. A composite question is processed by decomposing the original questions into atomic units, by either identifying the number of annotated concepts, utilizing conjunction tags (<IN>) and/or constraint expressions. *Table* 2 shows the variants of composite questions pattern. From the detected linguistic triple and the tagged POS the concepts are detected. These are then

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

manipulated to detect the information needs denoted by *info1* and *info2*.

Table 2: Composite Questions

Questions	Linguistic Triple	POS tag	Concept	Info1	Info2
What is the most populated state bordering oklahoma?	Most, populated, state, bordering, Oklahoma	JJS, JJ, NN, VBG, NN	P1, S1, P2, O1	state, bordering, Oklahoma	most populated< (?v1)
What is the lowest point of the state with the largest area?	Lowest point, state, largest area	JJS NN, JJS NN	S1, S2, P1	state, largest area	lowest point(?v1)
what are the populations of states through which the mississippi river runs?	Populations, states, missisippi river, runs	NN, NNS, JJ NN, VBZ	P1, S1, S2	state, mississippi river, runs	populations (?v1)

When dealing with SPARQL construction where multi-triples are involved, the following procedure should be executed:

1. Identify the linguistic triple
2. Identify the number of properties to be used, and the connected classes
3. Prioritize object property
4. Identify the variable to represent the l_o and the type of connection with the property
5. If focus=l_o
6. The same variable should represent the same class, despite its connection as domain or range

The challenge to automatically decompose the question into its sub-question increases when there are more than one aggregate expressions used, such as shown in Figure 3 where three triples are created so that each answers the sub-questions; (i) the smallest state, (ii) the capital of the smallest state, (iii) the population of the capital of the smallest state. Although the predicate can be identified from the question, challenge arises in arranging the triple according to their concepts connectivity. The *focus* of the question is also important because the aggregation operator is activated by the variable that represents the *focus.*

```
PREFIX geo:<http://www.mooney.net/geo#>
PREFIX xsd:
    <http://www.w3.org/2001/XMLSchema
    #>
PREFIX rdfs:
    <http://www.w3.org/2000/01/rdf-
    schema#>
PREFIX rdf:
    <http://www.w3.org/1999/02/22-rdf-
    syntax-ns#>
```

Figure 3: SPARQL query for 'what is the population of the capital of the smallest state?'

A deeper analysis of the question needs to be performed for questions that contain comparative expression. Figure 4 shows the SPARQL syntax for the question 'which states have points higher than the highest point in Colorado?' where there are two information needed*; info1*=highest point in Colorado; and *info2*= states have points higher than *info1*. Therefore the identification of the observational linguistic triples should be done by first breaking the questions and then merging them according to the connecting property, in this case the comparison of variable *?e* and *?z* while the focus is *?s*.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

```
PREFIX geo:<http://www.mooney.net/geo#>
PREFIX xsd:
      <http://www.w3.org/2001/XMLSchema
      #>
PREFIX rdfs:
      <http://www.w3.org/2000/01/rdf-
      schema#>
PREFIX rdf:
      <http://www.w3.org/1999/02/22-rdf-
      syntax-ns#>
PREFIX afn:
```

Figure 4: SPARQL Query For 'Which States Have Points Higher Than The Highest Point In Colorado?'

3.1.3 Answer type

The *answer type* indicates the form of generated answer; be it list, definition or affirmation. The user friendliness of a QA system can be increased by utilizing NL generation technique where complete grammatical sentence can be returned along as the answer from the reasoned SPARQL query.

3.1.4 Operation

The NL questions that require *negation* and arithmetic operations such as *comparative*, *superlative*, and *aggregation* can generally be classified by the occurrence of the relevant expressions. *Negation* can be represented by the expressions such as 'not', 'outside', and 'except' while the *comparative* operation can be detected by the occurrence of the tagged part-of-speech pattern <JJR><IN> which parses fragments such as 'larger than', while the identification of *superlative* expression is by detecting the <JJS> tag.

The *superlative* verb lexicons such as 'largest', 'smallest', 'greatest', and 'biggest' can be stored to represent its orientation i.e., the orientation of 'largest', 'greatest' and 'biggest' is descending while for 'smallest' is ascending. The superlative verb lexicons are paired with the matching ontology property. Question identifier such as 'How many' and 'How much' which is tagged by <WHADP> or <JJ> indicates a *quantification* expression that requires the 'COUNT' operation while the 'SUM' operation is indicated by expressions such as 'combination of' and 'sum of'.

3.2 Semantic Ambiguity in Input-Knowledge Base Mapping

The semantic ambiguity exists mainly due to the linguistic variance of terms used by the user in the NL question which may result to two issues; (i) indirect matching terms with the KB content and (ii) implicit needs of ontology concepts. Both issues affect the relevance of generated SPARQL query and returned answer because correct translation really depends on whether the information is represented in the KB.

In order to solve the first issue standard KB naming should be practised to make sure the translated SPARQL query is very close to the original NL question. This is a pressing problem especially in the concept annotation task which is among the crucial process in the NL-to-SPARQL translation because it provides information on the type of the ontology concepts required in the question. String matching formula and WordNet can also be used to obtain the most similar matching concepts. Clarification dialogues [9,11] has also been used where the user choose the recommended ontology concepts. For example, in the question 'How many citizens live in Alabama?' where the detected property 'live' has no direct match in the ontology. The correct concept to use in this case is 'statePopulation'.

The second issues relates to ambiguous KB concept issue when in the NL question there is no occurrence of concept property so the NL-to-SPARQL translation model needs to disambiguate the necessary concepts which may lead to more multiple properties or multiple classes. For example in the question 'what is the longest river in the US?' the detected linguistic triple is <river, ?, ?> where the correct property name, 'length' cannot be detected directly and in 'What state has the highest elevation?' where the correct properties are 'isHighestPointOf' and 'hiElevation'. In this situation there are two are possible solutions; utilizing an internal vocabulary or interacting with the user to choose closest matched property through clarification dialogue. Besides this, increased automation can be provided by loading the properties shared by the question *focus* and the detected linguistic triples.

4. TRANSLATING NATURAL LANGUAGE QUESTION TO SPARQL QUERY FOR SEMANTIC QUESTION ANSWERING

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

The NL-SPARQL translation process consists of several steps starting from the identification of variables in the NL question, mapping the variables with the ontology concepts and arranging the mapped variables in a logical SPARQL syntax. The basic variables are the linguistic triple which is a set of potential ontology concepts identified from the tokenized and parsed NL question. However, this information is insufficient especially when more complex question patterns are input. This is also the main factor of NLI limitation to date. Therefore, in this section we present the variables and rules for NL to SPARQL translation.

4.1 Variables for Natural Language Question to SPARQL Query Translation

We use the Stanford DependenSee parser to transform the NL into their part-of-speech constituents. Then we match each of the noun phrases in the branch with the ontology concept and perform concept annotation. For example, in the question 'How many states does the Colorado river run through?' taken from the Mooney's Geography NL query, 'state' and 'river' are annotated as class, 'Colorado' as instance of river and 'runsThrough' as object property. These are used to identify the linguistic triple based on the sequence of detected concept. This will generate three variables namely *identifiedSubject* (l_s), *identifiedPredicate* (l_p) and *identifiedObject* (l_o). The linguistic triples is <state, Colorado, run> while the *focus* is 'state'. The linguistic triples are then disambiguated to obtain the matching ontology triples resulting to t_s, t_o and t_p. However, this information is insufficient to capture the whole expression in the original question and to translate to an equivalent SPARQL query.

We introduce eight observational variables to broaden the translator coverage namely *classOfFocus* (c_f), *classTypeOfObject* (c_o) and *classTypeOfSubject* (c_s) identified by finding the class of the closest matching namespace of the *focus*, l_s and l_o in the ontology. The arithmetic typed questions are benefited from the *hasSuperlativeExp* and *orientationIs* variables which represent the orientation of the expression which is useful to know the type of sorting function to be inserted in the SPARQL syntax. The *hasCount* (example expression is 'how many') and *hasSum* (example expression is 'total of') are introduced to represent expressions indicating aggregation function.

However, the l_s, l_p,l_o, c_s, c_o, *focus, hasSuperlativeExp, orientationIs*, *hasCount* and *hasSum* variables are insufficient to process the translation of composite questions. These variables are useful when the ontology concepts can be mapped directly with the identified variables and they are limited to question that is translated to single triple (and with up to 3 variables) only. This is because these variables neglect the representation of multiple ontology concepts and their connections. For example in the question 'which states have points higher than the highest point in colorado?', these variables can only represent one property namely 'isHighestPointOf' and missed out 'hiElevation'.

Therefore we introduce *isCompositeQuestions* which is assigned a Boolean value of 'true' when more than one conjunction is detected and when more than two types of concepts are detected in the question. The *hasComparativeExp* variable is assigned a Boolean value of 'true' when the 'than' keyword is detected in the question. The comparative values are also observed to be assigned in the SPARQL query. The *hasNegationExp* variable is introduced to denote that the question contains negation expression. The parameters that involve negation are tracked to be assigned in the SPARQL query. In the next section we present the issues related to the automatic translation of composite NL question.

4.2 Rules for Natural Language Question to SPARQL Query Translation

4.2.1 Natural language questions linguistic pattern

In this section we present twelve linguistic patterns of NL. The linguistic-based question understanding depends on the patterns In each case the characteristics of each pattern is described followed by the template that comply with the pattern (shown in the first column in every tables), followed by example of compliant question. The section is divided into two sections; for single triple and multi-triples.

4.2.1.1 Single triple.

Pattern 1-the focus is a property value and controlled by the sorting operation

In Pattern 1 (Table 3), the *focus* the question relates to a property value such as 'What is the

Journal of Theoretical and Applied Information Technology
31st October 2015. Vol.80. No.3

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

area of the largest state?', where 'area' can be considered as an object property. The l_s, l_p, and c_s are known while l_o is unknown. This type of question is represented by the following template as SPARQL statement. The ORDER BY DESC operator is used to filter the answer to this question.

Pattern 2- focus is a class, and the answer is controlled by specified property and subject.
Pattern 2 illustrates the linguistic pattern of NL question that has a class as its focus and the answer is controlled by specified property and subject, as shown in *Table 4*. For example, in 'What is the capital of Texas?', a specific instance, l_o=texas can be identified. The l_p in the triple is replaced by highest matching score loaded based on the properties connected to the c_o and the l_p detected in the question. Since in this example the *focus* is a class and it is the domain of the l_p, *focus* is positioned as the subject of the triple.

Pattern 3-Focus is a class and the answer is controlled by the operation of the property.
Since only one class is detected and a numeric expression is detected with orientation ascending, the datatype properties that are connected to the class are loaded. The closest matched property to the l_p is identified by executing similarity matching function. The suitable property (t_p) is detected by *lo*ading the properties connected to the l_s. The variables in the SPARQL are (i) the answer, denoted by focus (*?foc*); in this example the *focus* is equal to the subject and (ii) the assignment of the range of the datatype property. *Table 5* shows the translation template for Pattern 3.

Pattern 4-Aggregate function .
An aggregate function is detected by the phrase 'how many'. The *focus* of the question is defined by the concept that needs to be aggregated. The rest of the SPARQL construction is performed by identifying the l_s, l_p and l_o. In the example shown follows 4 linguistic triples are detected which are 'state, colorado, river, run through'. The *focus* of the question is 'state' and a specific instance for the 'river' object is given, namely 'colorado'. Therefore, to compute the aggregate function for the state, the namespace of 'colorado' as a river is obtained and inserted in the SPARQL query together with the property that matches the l_p with the highest score (*Table 6*)

.

Table 3: Template for Pattern 1

Template	Translated SPARQL
SELECT *?foc* WHERE {*?foc* :t_p *?v0.*} ORDER BY ASC (xsd:float(*?v0*)) LIMIT 1	SELECT *?foc* WHERE {*?foc* :stateArea *?v0.*} ORDER BY DESC(*?v0*) LIMIT 1

Table 4: Template for Pattern 2

Template	Example	Translated SPARQL
SELECT *?foc* WHERE {*?foc* :t_p :t_o.}	What is the capital of Texas?	SELECT *?foc* WHERE {*?foc* :isCapitalOf :texas .}

Table 5: Template for Pattern 3

Template	Example	Translated SPARQL
SELECT *?foc* WHERE {*?foc* :t_p *?v0* .} ORDER BY ASC (xsd:float(*?v0*)) LIMIT 1	Which state has the smallest population density?	SELECT *?foc* WHERE {*?foc* :statePopDensity *?v0* .} ORDER BY ASC(xsd:float(*?v0*)) LIMIT 1

Table 6: Template for Pattern 4

Template	Example	Translated SPARQL
SELECT DISTINCT (COUNT(*) AS ?num) WHERE {:c_s :t_p *?foc* .}	How many states does the Colorado river run through?	SELECT DISTINCT (COUNT(*) AS ?numr) WHERE {:colorado2 :runsThrough *?v0* .}

Pattern 5-Aggregation.

In this pattern the *focus* is the property, and aggregate function 'SUM' (detected by the keyword 'combined') is performed on the l_p. The

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

linguistic triples detected are l_p=area and l_s=state. *Table 7* shows the template for Pattern 5.

Pattern 6-Ambiguity of object name.

In this pattern the *focus* is a datatype property value. The correct disambiguated linguistic triple is l_o=mississippi and l_p=length. The c_o=state with the namespace 'mississippi2'. Therefore the triple is arranged by highlighting the value of the 'length' as the answer type, as shown in *Table 7*.

Table 7: Template for Pattern 5

Template	Example	Translated SPARQL
SELECT *?foc* (SUM(xsd:float(*?v0*)) AS ?total) WHERE {*?foc* :t_p *?v0* ; :t_p *?v0*. }	What is the combined area of all 50 states?	SELECT *?v0* (SUM (xsd:float(*?v0*)) AS ?total) WHERE {*?foc* :stateArea *?v0* ; :stateArea *?v0*. }

Table 8: Template for Pattern 6

Template	Example	Translated SPARQL
SELECT *?foc* WHERE {l_s :l_p *?foc*.}	What length is the Mississippi?	SELECT *?foc* WHERE {:mississippi2 :length *?foc*.}

Table 9: Template for Pattern 7

Template	Example	Translated SPARQL
SELECT *?foc* WHERE{ *?foc* :t_{p1} :t_o . *?foc* :t_{p2} *?v0*.} ORDER BY DESC (xsd:float(*?v0*)) LIMIT 1	What is the longest river in Texas	SELECT *?foc* WHERE{ *?foc* :runsThrough :texas . *?foc* :length *?v0*.} ORDER BY DESC (xsd:float(*?v0*)) LIMIT 1

4.2.2.2 Multi-triples

Pattern 7-Focus is a class and the answer is controlled by the operation on the property connected with a specified subject.

An implicit t_p is detected and added to the SPARQL by loading the concepts connected with the class of specified subject. In this pattern only l_s and l_o are detected. The correct properties cannot be detected directly from the linguistic triple. In the example l_s=river and t_o=texas, therefore c_s=river and c_o=state. The question also contains a numerical expression with orientation ascending. Because of this, the datatype properties connected to the c_s is loaded, leading to the identification of t_p='length'. However, the 'length' property is connected to 'river' as its domain and 'float' as its range. Since 'texas' is not represented as an instance of the 'river' class, the object properties that is connected to 'river' as a domain are loaded, leading to 'runThrough' and 'hasRiver'. However, in the ontology used as example, 'hasRiver' is ignored because it is never used in the ontology; instead, 'runThrough' which is its inverse is used majorly. Therefore, the 'length' is t_{p1} and another triple is created that use 'runThrough' as its property (t_{p2}), and 'texas' is assigned as the to; where the triple is <*?foc* :runsThrough :texas> and the second is <*?foc* :length *?v1*>. The *?v0* variable is to denote the focus of the question while *?v1* denotes the length of the river. The 'ORDER BY' and 'LIMIT 1' operation are also appended to allow the computation of the longest *?v1* as shown in Table 9.

Pattern 8-Identification of implicit property and no specified subject.

The processing required for this pattern is similar to the pattern in Pattern 5. In the example the *focus* is a class and l_s=mountain and the answer

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

is controlled by the operation performed on the value of datatype property. Therefore the datatype property that is connected to 'mountain' is loaded, which is 'height'. However, since the answer requires a class, the object property that is linked to the class of focus is loaded, in this example 'isMountainOf'. Table 10shows the template for Pattern 8.

Pattern 9-Identification of implicit class and property.

This pattern illustrates usage of two classes; one explicit and another implicit. The *focus* is 'state' and the answer is controlled by 'the highest elevation'. The detected linguistic triples is l_s=state and l_p=hiElevation. Although the usage of 'hiElevation' as the property to be used complies with the previous analysed patterns, here the answer type cannot be detected from the triple generated from this property only. The challenge in this pattern is that the method used in the previous patterns is not suitable because the c_s is not connected directly to l_p. Instead, two triples are needed where the domain of the most matching property name with l_p is 'HiPoint'. The 'HiPoint' class is connected to the 'State' class by an object property called 'isHighestPoint'. Therefore, the first connecting triple that utilizes 'isHighestPoint' is added in the SPARQL query (Table 11); and the algorithm should be generalized to extend into the detection of the connecting triple.

Pattern 10-Composite question.

The *focus* in this pattern is a property. This pattern as shown in Table 12 is an example of a composite query because multi-triple is involved. The detected linguistic triples are l_{p1}=population, l_{p2}=capital and l_s=state. However, this information is insufficient to aid in building the equivalent SPARQL query to the original question. The question requires three triples: w=ASC(stateArea(x)) LIMIT 1, y=isCapitalOf(x), z=statePopulation(x) where x denotes the 'state', w denotes the stateArea of x, y denotes the 'capital of x' and z denotes the 'population of state x'. To compute this question the following steps should be used: (i) create a triple based on the detected predicates which generates <?v_0 :statePopulation ?v_1> and <?v_2 :isCapitalOf ?v_0> (ii) create a triple that answers the 'smallest state' expression will generates <?v_0 :stateArea ?v_3 ORDER BY (DESC(?v_3))>.

Table 10: Template for Pattern 8

Template	Example	Translated SPARQL
SELECT *?foc* WHERE { *?foc* :t_{p1} *?v*$_0$. *?foc* : t_{p2} *?v*$_1$. } ORDER BY DESC (xsd:float(*?v*$_1$)) LIMIT 1	What is the highest mountain in the US?	SELECT *?foc* WHERE { *?foc* :isMountainOf *?v*$_0$. *?foc* :height *?v*$_1$. } ORDER BY DESC (xsd:float(*?v*$_1$)) LIMIT 1

Table 11: Template for Pattern 9

Template	Example	Translated SPARQL
SELECT *?foc* WHERE { *?v*$_0$:pred1 *?foc*. *?v*$_0$:pred2 *?v*$_1$. } ORDER BY DESC (xsd:float(*?v*$_1$)) LIMIT 1	What state has the highest elevation?	SELECT *?foc* WHERE { *?v*$_0$:isHighestPointOf *?foc*. *?v*$_0$:hiElevation *?v*$_1$. } ORDER BY DESC (xsd:float(*?v*$_1$)) LIMIT 1

Table 12: Template for Pattern 10

Template	Example	Translated SPARQL
SELECT *?foc* WHERE { *?v*$_0$ *lp*$_1$ *?v*$_1$. *?v*$_2$ *lp*$_2$ *?v*$_0$. *?v*$_2$:*lp*$_3$ *?foc* . ORDER BY ASC (xsd:float(*?v*$_2$)) LIMIT 1	What is the population of the capital of the smallest state?	SELECT *?foc* WHERE { *?v*$_0$:stateArea *?v*$_1$. *?v*$_2$:isCapitalOf *?v*$_0$. *?v*$_2$:cityPopulation *?foc* . ORDER BY ASC (xsd:float(*?v*$_1$)) LIMIT 1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Pattern 11-Comparative operation.

The pattern is another example of composite query but more challenging than the previous pattern because it involves the combination of constraints and has several multi-triples and multi-variables. From the question the detected linguistic triples are l_{s1}=state, l_{s2}=point and l_o=colorado. However, these linguistic triples are insufficient to translate the question into its equivalent SPARQL, mainly because the question demands a constraint operator, in this example the expression indicating the necessity of constraint is 'higher than' and there is a specification to perform comparison given by the expression 'the highest point in colorado'. The question requires three triples: x=isHighestPointOf(colorado) which represents the expression 'the highest point in colorado', y=hiElevation(x) which represents the expression 'height of the highest point in colorado'; but cannot be detected directly from the question, b=isHighestPointOf(a) and c=hiElevation(b) where a denotes the 'state', b denotes the highest point of a state, a, while c denotes the 'high elevation (height) of b'. A comparative function namely 'FILTER' is used which can compare the value of c and y so that the correct answer can be computed. Table 13 shows the template of Pattern 11 translation.

Pattern 12-Negation.

This pattern illustrates negation expression in the NL question, which can be identified by 'do not'. To process this question the negation expression is first removed from the question and the equivalent SPARQL is constructed. Then, the term that is connected to the negation operator, in this example 'lake' is detected. The variable that represents 'lake' is also identified. The negation statement is then inserted in the form of 'FILTER(!bound(negationVar))'; as in Table 14.

The summarised linguistic patterns are shown in *Table 15*.

Table 13: Template for Pattern 11

Template	Example	Translated SPARQL

SELECT *?foc*
WHERE {
?v0 :tp1 :to.
?v0 :tp2 *?v1.*
OPTIONAL {
?v2 :tp1 ?foc.
?v2 :tp1 ?v3.
FILTER (*?v2* >*?v1*)}
	Which states have points higher than the highest point in Colorado?	SELECT *?foc*
		WHERE {
		?v0 :isHighestPointOf :colorado.
		?v0 :hiElevation *?v1.*
		OPTIONAL {
		?v2 :isHighestPointOf *?foc.*
		?v2 :hiElevation *?v3.*
		FILTER (*?v3* >*?v1*)}}}

Table 14: Template for Pattern 12

Template	Example	Translated SPARQL
SELECT *?foc* WHERE { *?foc :tp1 ?v0.* OPTIONAL{*?v1 :tp2 ?v0.*} FILTER (!bound(*?v1*))	Show the bordering states of the states that do not have any lake	SELECT *?foc* WHERE { *?foc* :borders *?v0.* OPTIONAL{*?v1* :isLakeOf *?v0.*} FILTER (!bound(*?v1*))

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Table 15: Linguistic Pattern Analysis

Case	Focus = subject	Focus concept	#subject	#property	#object	Object is instance	Has superlative expression	OrientationIs	Has Aggregate Expression	Has Comparative Expression	Number of atomic units	Has negation expression	Negation variable	Template
1.	YES	P	1	1	1	NO	YES	DESC	NO	NO	1	NO	N/A	SELECT *?foc* WHERE {$?v_1$:t_p *?foc*.} ORDER BY ASC (xsd:float(*?foc*)) LIMIT 1
2.	NO	C	1	1	1	YES	NO	N/A	NO	NO	1	NO	N/A	SELECT *?foc* WHERE {*?foc* :t_p :t_o.}
3.	YES	C	1	1	0	N/A	YES	DESC	NO	NO	1	NO	N/A	SELECT *?foc* WHERE {*?foc* :t_p $?v_0$.} ORDER BY ASC (xsd:float($?v_0$)) LIMIT 1
4.	YES	C	1	1	1	YES	NO	N/A	YES	NO	1	NO	N/A	SELECT DISTINCT (COUNT(*) AS ?num) WHERE {:c_s :t_p *?foc* .}
5.	YES	P	1	1	0	NO	NO	N/A	YES	NO	1	NO	N/A	SELECT *?foc* (SUM (xsd:float($?v_0$)) AS ?tot) WHERE{*?foc* :t_p $?v_0$; :t_p $?v_0$. }
6.	NO	P	1	1	1	YES	NO	N/A	NO	NO	1	NO	N/A	SELECT $?v_1$ WHERE {l_s :t_{p1} $?v_1$.}
7.	YES	C	1	2	1	YES	YES	ASC	NO	NO	1	NO	N/A	SELECT *?foc* WHERE {*?foc* :t_{p1} :t_o . *?foc* :t_{p2} $?v_0$.} ORDER BY DESC (xsd:float($?v_0$)) LIMIT 1
8.	YES	C	2	2	0	N/A	YES	ASC	NO	NO	1	NO	N/A	SELECT *?foc* WHERE {*?foc* :t_{p1} $?v_0$. *?foc* : t_{p2} $?v_1$.}ORDER BY DESC (xsd:float($?v_1$)) LIMIT 1
9.	NO	C	2	2	0	N/A	YES	ASC	NO	NO	1	NO	N/A	SELECT *?foc* WHERE {$?v_0$:t_{p1} *?foc*. $?v_0$:t_{p2} $?v_1$. }ORDER BY DESC(xsd:float ($?v_1$)) LIMIT 1
10.	NO	P	2	3	3	NO	YES	DESC	NO	NO	3	NO	N/A	SELECT *?foc* WHERE {$?v_0$:t_{p1} $?v_1$. $?v_2$:t_{p2} $?v_0$. $?v_2$:t_{p3} *?foc* .} ORDER BY ASC (xsd:float($?v_2$)) LIMIT 1
11.	YES	C	1	1	2	YES	YES	N/A	NO	YES	2	NO	N/A	SELECT *?foc* WHERE {$?v_0$:t_{p1} :t_o. $?v_0$:t_{p2} $?v_1$. OPTIONAL {$?v_2$:t_{p1} *?foc*. $?v_2$:t_{p1} $?v_3$. FILTER ($?v_2$ >$?v_1$)}
12.	YES	C	2	2	1	NO	NO	N/A	NO	NO	2	NO	*?v*1	SELECT *?foc* WHERE {*?foc* :t_{p1} $?v_0$. OPTIONAL {$?v_1$:t_{p2} $?v_0$.} FILTER (!bound($?v_1$))

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

4.2 Translation Model for Natural Language Question to SPARQL Query

A NL-to-SPARQL translator called MYAutoSPARQL is built which is comprised of three processes namely preprocessing, variables initialisation and semantic mapping. Figure 5 shows MYAutoSPARQL architecture. The rules for the translator is shown in Figure 6 and the template for the SPARQL is shown in *Table 16*.

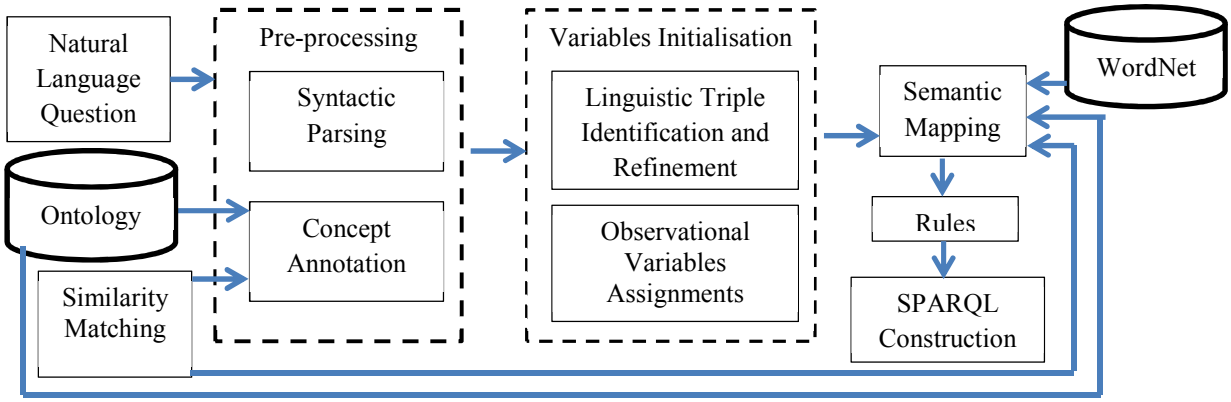

Figure 5. MYAutoSPARQL Architecture

1. If *focus* IS object property value or *focus* IS Class
   a. If l_s==class name
      i. If l_o==null
         1.If Edge(l_p,Focus)=1 Case 1
         2.Else Load properties shared between *focus* and l_s. Case 2
      ii. Else If l_o!=null Case 2
   b. Else
      i. If t_p IS object property
         1.IF t_p is domain of t_p Case 3
         2.ELSE Case 4
      ii. End if
   c. End If
2. Else if *focus* IS datatype property value
   a. If l_s==null Case 5
   b. Else Case 6
3. Else if *focus* IS (SUP) Datatype property Val Case SupDP
4. Else if *focus* IS (SUP) Class
   a. If t_p==null and l_o!=null
      i. Load shared properties between *focus* and l_o
      ii. Case SupClass
   b. End If
5. Else if *focus* IS (COUNT) Class Case CountClass
6. Else if *focus* IS (SUM) Datatype property Val Case SumDP
7. End IF

Figure 6: NL-to-SPARQL Translation Rules

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Table 16: SPARQL Template for the Linguistic Cases in Figure 6

Case 1	SELECT ?foc WHERE ?foc :tp ?v0 .
Case 2	SELECT ?foc WHERE ?foc :tp :to .
Case 3	SELECT ?foc WHERE $:c_s$ $:t_p$ *?foc* .
Case 4	SELECT ?foc WHERE ?foc $:t_p$ $:c_s$.
Case 5	SELECT ?v0 WHERE ?v0 :tp ?foc .
Case 6	SELECT ?v0 WHERE ?:s :tp ?foc .
Case SupDp	SELECT ?foc WHERE ?v0 :tp ?foc. ORDER BY DESC(?foc)
Case SupClass	SELECT ?foc WHERE ?foc :tp1 :lo. ?foc :tp2 :v0. ORDER BY DESC(?v0)
Case CountClass	SELECT ?foc with SELECT DISTICT COUNT(*) AS ?num
Case SumDp	SELECT *?foc* (SUM(xsd:float($?v_0$)) AS ?total) WHERE {*?foc* $:t_p$ $?v_0$; $:t_p$ $?v_0$. }

5. EVALUATION AND RESULTS

An experiment is carried out to measure the effectiveness of implementing the set of observational variables in processing the semantic questions through MYAutoSPARQL. The FREyA[1] system is used as a benchmark application. It is chosen because FREyA uses the same dataset for their development and testing. It is also easily accessed online.

The questions are split into three levels of complexities; selection (106 questions), arithmetic (64 questions) and composite (20 questions). A QA system should be evaluated based on the difficulty of questions that it can handle, besides the standard accuracy evaluation [19]. Furthermore, accurate answer to a question can only be produced given several circumstances; (i) the question is understood well and (ii) answer is available in the referral source.

5.1 Performance of Observational Variables

We first evaluate the performance of each of the observational variables. Table 17 shows the correctness score of each linguistic variable on different question complexity. Since there is no existing study that focuses on similar depth of linguistic analysis, evaluation is performed by comparing the result with expert judgment. In each cell the value is given by n/m where n denotes the correct extraction while m is the total number of occurrence of the variable in the question. N/A indicates that the variable does not exist in the question complexity. The *isCompositeQuestions*, *hasComparative*, and *hasNegation* are not observed because naturally they only exist in composite typed questions. Therefore, the *isCompositeQuestions* is always TRUE in this case, while the only *hasComparative* appears only in 1 case and no example of *hasNegation* is provided in the adopted dataset

The performance of the linguistic variables execution on the selection is the best followed by arithmetic and composite. The *hasSuperlativeExp* and *orientationIs* have perfect score because of the coverage effectiveness of using expression based identification. Among the triples identification, l_s has the highest performance and seconded by l_o. This is because the disambiguation challenge in l_p is much higher due to implicit expression in the question and low similarity score between the l_p and the property name in the KB. The identification of *focus*, c_s and c_o is quite satisfying. However, the major observation from this experiment is that the usage of the linguistic variables does not guarantee correctness of translated SPARQL especially in the composite question processing. This is because constructing the SPARQL requires adoption of several rules and connecting variables identification.

5.2 Performance in Clarification Dialogues Reduction

The measurement parameters used to evaluate the performance in are (i) correct, to indicate accurate answer is generated by the system, (ii) partial correct indicates only part of the answer is correct, (iii) wrong means the answer is incorrect and (iv) failed score is counted based upon unsuccessful (no) action performed by the system. Since FREyA utilizes clarification dialogue and MYAutoSPARQL aims to eliminate user intervention in the answer generation, the number of dialogues involved is observed.

The comparison of FREyA against MYAutoSPARQL using the selection-typed questions and the score is shown in

[1] http://services.gate.ac.uk/freya/

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Table 18. Besides higher score in terms of the *Correct* and *Fail* computation, MYAutoSPARQL has managed to eliminate 73.6% of clarification dialogues executed in FREyA. It is shown that MYAutoSPARQL achievement of 74.5% automation and correct translation is far better compared to FREyA's ability to answer 28 (26.4%) questions correctly without any clarification dialogue. It is also discovered that MYAutoSPARQL 's failed processed queries are smaller (13 questions) compared to FREyA's 21 questions. However, when MYAutoSPARQL has processed 9 queries wrongly, FREyA has managed to answer 7 of them, 1 without any clarification dialogue and 6 with 1 dialogue. The 13 questions failed to be processed by MYAutoSPARQL has been answered by FREyA; 2 questions automatically, 3 by involving 1 dialogue and 1 by involving 3 dialogues while another 7 is failed to be processed.

Table 19 shows the comparison of score between MYAutoSPARQL and FREyA in processing arithmetic-typed questions. MYAutoSPARQL has improved twice the performance of FREyA although the number of failed processing in MYAutoSPARQL is slightly larger than FREyA. The high score is mainly aided by the introduced observational variables. However, the lower score is caused by the weakness in mapping the correct property name.

.

Table 17: Result On Correctness Score Of Observational Variables

Question Complexity	Linguistic Variables										*SPARQL*
	l_s	l_p	l_o	*focus*	c_s	c_o	c_f	*hasSuperlativeExp*	*orientationIs*	*hasAggregateExp*	
Selection	106/106	84/106	93/106	106/106	106/106	93/106	106/106	N/A	N/A	N/A	79/106
Arithmetic	50/64	23/64	52/64	64/64	64/64	50/64	64/64	47/47	47/47	11/13	33/64
Composite	21/33	11/35	20/30	8/20	21/33	17/30	8/20	1/1	19/19	0/1	1/20

Table 18: Statistics Of Clarification Dialogues Used In Freya For Selection-Typed Questions

Parameter	**Score**		**Clarification Dialogues in FREyA**			
	MYAutoSPARQL	**FREyA**	**0 Dialogue**	**1 Dialogue**	**2 Dialogues**	**3 Dialogues**
Correct	79	73	24	30	11	8
Partial Correct	5	5	1	2	2	0
Wrong	9	7	1	6	0	0
Fail	13	21	17	3	0	1
Total	106	106	43	41	13	9

Table 19: Comparison Between Myautosparql And Freya For Arithmetic-Typed Questions

Parameter	**MYAutoSPARQL**	**%**	**FREyA**	**%**
Correct	33	51.56	17	26.56
Partial Correct	0	0.00	16	25.00
Wrong	15	23.44	18	28.13
Fail	16	25.00	13	20.31
Total	64	1	64	1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Table 20: Statistics Of Clarification Dialogues Used In Freya For Arithmetic-Typed Questions

Parameter	0 Dialogue	1 Dialogue	2 Dialogues	3 Dialogues	4 Dialogues
Correct	0	10	7	0	0
Partial Correct	0	1	15	0	0
Wrong	1	5	4	5	3
Fail	1	5	6	1	0
Total	2	21	32	6	3

Table 20 shows the statistics of clarification dialogues used in FREyA for arithmetic-typed questions answering. FREyA has executed repeatedly dialogues to clarify the superlative expressions (e.g, smallest, lowest, etc.) to be related with constraint operations such as 'max' and 'min'. This is burdening the user and FREyA should have adopted a method to automatically disambiguate the orientation of the superlative expressions with the suitable constraint operation.

Since FREyA could not process composite question no comparison is provided against MYAutoSPARQL . Part of the reason of FREyA's failure is no ability to process composite queries, where it demands more than one property. Furthermore, their existing strategy to disambiguate is solely depending on the identification of superlative expressions. For example in the question 'what is the shortest river?', the first clarification dialogue will ask the user to clarify the term 'shortest', with all properties connected to 'river' is loaded. The closest property is either 'runsThrough' or 'length'. However, the correct processing requires both properties to be executed.

6. DISCUSSION AND CONCLUSION

The fundamental steps in the translation of an NL question into a SPARQL query are identification of linguistic triples, mapping of linguistic triples with ontology concept and construction of the SPARQL. A lot of disambiguities may arise but the focus of the paper is instead on the translation process. We argue that linguistic triples are insufficient to represent the variants of linguistic patterns posed due to the inherited dynamicity in NL question formulation.

Our preliminary work on template-based translation for arithmetic question processing has shown that it is only efficient when single triple is involved in the translated SPARQL syntax. Although this is an advancement judging from the current approaches available in arithmetic question processing, we discover more composite patterns which require deeper processing. In this paper we highlight a model for NL question processing for semantic search where equivalent SPARQL query is constructed. The MYAutoSPARQL model covers multi triples and multi variables which have not been processed by any of the existing technique in NL-SPARQL translation. On the contrary, current approach mainly focuses on the ambiguity reduction in semantic mapping by using the clarification dialogue; however this may be confusing and does not promise accurate translation.

In this paper twelve linguistic patterns are identified which are distinguished based on the type of *focus*, types of constraint modifier, occurrence of aggregate expression, negation typed questions and composite questions. We introduce 10 observational variables namely *focus*, c_f, c_o, c_s, *hasSuperlativeExp, orientationIs*, *hasAggregateExp*, *isCompositeQuestions, hasComparativeExp, hasNegationExp* which can aid in tracking the information needs in the question. These variables also indicate the kind of operations needed in processing the question.

MYAutoSPARQL is compared against FREyA with two objectives; (i) effectiveness of the observational variables, (ii) competency of NLI system without clarification dialogue component. The results have shown that the observational variables have increased the processing ability and the elimination of clarification dialogue has not influence the performance. MYAutoSPARQL has shown comparable, if not better achievement compared to FREyA in selection and arithmetic question processing. Although the composite question processing has not reached much accomplishment, the observational variables have improved the *understanding* of the questions, towards better processing ability. This is also closer to human's thinking in decomposing the

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

question into its sub-questions and federating the final answer based on the connecting variables.

The suggested future work is on better computation for composite questions, adaptive semantic disambiguation techniques testing on other dataset. This area should be studied by more researchers towards higher system intelligence.

7. ACKNOWLEDGEMENTS

This paper reports the post-doctoral study as part of the national level project conducted in Malaysia entitled "Logical Semantic Structured Language Model for Knowledge-Based Society" (LRGS/TD/2011/UITM/ICT/03) sponsored by the Ministry of Education, Malaysia. I would like to thank the sponsor for the encouragement and to the project members for their cooperation on this work especially the members from the University Putra Malaysia (UPM). I would like to also extend my gratitude to the proof-readers and reviewers of this paper Associate Professor Dr. Masry Ayub and Dr. Masnizah Mohd from the Faculty of Technology and Information Science, National University of Malaysia. I owe an appreciation to several postgraduate researchers at the Faculty of Computer Science and Information Technology, UPM for numerous academic discussions on natural language interface and semantic search. Thank you for all your kind help.

REFERENCES

[1] Q.Q. Guo, M. Zhang, Question answering based on pervasive agent ontology and Semantic Web, Knowledge-Based Systems. 22 (2009) 443–448.

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

[3] Q. Zhou, C. Wang, M. Xiong, H. Wang, Y. Yu, SPARK: Adapting Keyword Query to Semantic, in: Proceedings of the 6th International The Semantic Web and 2nd Asian Conference on Asian Semantic Web Conference, 2007: pp. 694–707.

[4] E. Kaufmann, A. Bernstein, How useful are natural language interfaces to the semantic web for casual end-users?, in: The Semantic Web, 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference, 2007: pp. 281–294.

[5] N.M. Sharef, S.A. Mohd Noah, Natural Language Query Translation for Semantic Search, Journal of Digital Content, Technology and Applications. (2013) in press.

[6] P. Cimiano, P. Haase, M. Mantel, ORAKEL: A Portable Natural Language Interface to Knowledge Bases Introduction, 2007.

[7] P. Cimiano, P. Haase, J. Heizmann, M. Mantel, R. Studer, Towards portable natural language interfaces to knowledge bases – The case of the ORAKEL system, Data & Knowledge Engineering. 65 (2008) 325–354.

[8] D. Damljanovic, M. Agatonovic, H. Cunningham, Natural Language Interfaces to Ontologies : Combining Syntactic Analysis and Ontology-based Lookup through the User Interaction, in: Proceedings of the 7th Extended Semantic Web Conference (ESWC 2010), 2010: pp. 1–15.

[9] V. Lopez, M. Fernández, E. Motta, N. Stieler, PowerAqua: Supporting Users in Querying and Exploring the Semantic Web, Semantic Web. 3 (2012) 249–265.

[10] D. Damljanovic, M. Agatonovic, H. Cunningham, R. Court, P. Street, Identification of the Question Focus: Combining Syntactic Analysis and Ontology-based Lookup through the User Interaction, in: Proceedings of the International Conference on Language Resources and Evaluation (LREC 2010), 2010.

[11] D. Damljanovic, M. Agatonovic, H. Cunningham, FREyA: an Interactive Way of Querying Linked Data Using Natural Language, in: Proceedings of the 8th International Conference on The Semantic Web, 2011: pp. 125–138.

[12] A. Hogan, A. Harth, J. Umbrich, S. Kinsella, A. Polleres, S. Decker, Searching and browsing Linked Data with SWSE: The Semantic Web Search Engine, Web Semantics: Science, Services and Agents on the World Wide Web. 9 (2011) 365–401.

[13] E. Kaufmann, A. Bernstein, R. Zumstein, Querix: A Natural Language Interface to Query Ontologies Based on Clarification Dialogs, in: Proceedings of the 5th

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

International Semantic Web Conference (ISWC 2006), 2006: pp. 980–981.

[14] E. Kaufmann, A. Bernstein, L. Fischer, NLP-Reduce: A naive but domain-independent natural language interface for querying ontologies, Proceedings of the European SemanticWeb Conference ESWC 2007. (2007).

[15] Y. Lei, V. Uren, E. Motta, SemSearch : A Search Engine for the Semantic Web, (2006) 238–245.

[16] C. Unger, L. Bühmann, J. Lehmann, A.-C. Ngonga Ngomo, D. Gerber, P. Cimiano, Template-based question answering over RDF data, Proceedings of the 21st International Conference on World Wide Web - WWW '12. (2012) 639.

[17] M. Yahya, K. Berberich, S. Elbassuoni, M. Ramanath, V. Tresp, G. Weikum, Deep answers for naturally asked questions on the web of data, Proceedings of the 21st International Conference Companion on World Wide Web - WWW '12 Companion. (2012) 445–448.

[18] L. Zemmouchi-ghomari, Translating Natural Language Competency Questions into SPARQLQueries: A Case Study, in: The First International Conference on Building and Exploring Web Based Environments, 2013: pp. 81–86.

[19] H.-J. Oh, K.-Y. Sung, M.-G. Jang, S.H. Myaeng, Compositional question answering: A divide and conquer approach, Information Processing & Management. 47 (2011) 808–824.

[20] M. Gao, J. Liu, N. Zhong, F. Chen, C. Liu, Semantic Mapping From Natural Language Questions To Owl Queries, Computational Intelligence. 27 (2011) 280–314.