Resubmission of SWJ # 956-2167,
Ripple Down Rules for Question Answering.

June 15, 2015

Contents

1 Author(s) cover letter responding to the original reviews

Starts on next page.

Dear SWJ Editors-in-Chief,

We would like to submit the revised manuscript entitled "Ripple Down Rules for Question Answering" to the Special Issue of the Semantic Web Journal on Question Answering over Linked Data.

The guest editors, Prof. Christina Unger and Prof. Axel Ngonga, informed us of a “major revision” decision for our original manuscript (SWJ submission 956-2167). Based on the reviewers’ comments, we have made changes which are detailed as follows.

RESPONSE TO GUEST EDITORS:

We have added materials to address issues raised by the reviewers. The major changes are:

- *We have restructured section 4.3 to add a new subsection 4.3.3 “Porting to other domains”. This new subsection is to illustrate the process of extending an existing knowledge base to analyze DBpedia and biomedical test questions.*

- *We have added question analysis evaluation results on DBpedia and biomedical test questions in section 5.1.2.*

RESPONSE TO REVIEWER 1:

3.1. The query format used in this section should be explained more carefully. The example 'which univ does Pham Duc Dang study in' introduces a term1 'university' which seems to be a concept/category term. How is this handled in the automatic translation to SPARQL?

> *We have modified section 3.1 and appendixes A and B to clarify the format.*
>
> *KbQAS does not translate the intermediate representation of an input question to a SPARQL query. In KbQAS, the Ontology mapping module is responsible for finding elements of the target ontology, in which these elements are corresponding to the terms in the input question. We have modified section 3.4 to clarify this.*

3.1./3.2. The examples shown in these sections seem rather straightforward compared to questions in the most recent editions of QALD (www.sc.cit-ec.uni-bielefeld.de/qald/). The remarks at the end of section 5, concerning the impossibility of answering comparative questions (who has the highest GPA) also suggest that the scope of questions is rather limited. Please elaborate on this.

> *We have clarified section 5.2 to show that the impossibility of answering comparative questions comes from the answer retrieval component of KbQAS. In section 5.2, we present answering results to questions which are successfully analyzed by the question analysis component. The question analysis component of KbQAS enables to analyze comparative questions.*
>
> *For example, as shown in table 5, the Vietnamese and English knowledge bases for question analysis contain 5 and 8 structure patterns of comparative questions, respectively.*

3.3.2. 'a manual dictionary is built for describing concepts..in the ontology', and 'a phrase is matched by one of the relation patterns'. How much manual labor is involved in this, and how does it compare to the development of question patterns described later? In particular, do these steps not also make the translation process dependent on the ontology in undesirable ways?

> *We thank for the questions. We have improved section 3.3.2.*
>
> *The dictionary contains concepts which are (automatically) extracted from the (any) target ontology (or domain). However, there is no publicly available WordNet-like semantic lexicon for Vietnamese. So we manually add synonyms of the extracted concepts to the dictionary (This process is fast due to the small number of synonym concept pairs).*
>
> *As presented in section 3.3.2, the dictionary is only used to determine whether a noun phrase in Vietnamese is a concept or entity type, together with two other heuristics. It is not used to identify noun phrases. To indentify noun phrases and relation phrases in Vietnamese, we used five JAPE grammar patterns (It was quickly to develop these five patterns based on Vietnamese language grammars).*
>
> *Importantly, these steps do not make the translation process dependent on the target ontology in undesirable ways.*
>
> *The crucial task is to identify noun phrases, question phrases and relation phrases because these phrases are used to capture the question structure patterns. The concept or entity type information inside the noun phrases is helpful but not so important. If the concept or entity type information exists, it could somehow reduce the ambiguities in Vietnamese question patterns. Otherwise, we simply add exception rules into the knowledge base to handle ambiguity cases.*
>
> *Specifically, when applying our question analysis approach to English, we constructed a knowledge base for English question analysis without having any concept or entity type information available, as illustrated in section 4.3.*
>
> *(We reused the JAPE grammar patterns which AquaLog [27] had used to identify phrases in English, in which these patterns did not produce any information about concept or entity type.)*

4. rules for question analysis

It is hard to understand the details of this section, as it uses an idiosyncratic notation for queries. Is it possible to show the output of the matching process as (schematic) SPARQL queries?

> *We have clarified section 4. We believe it is possible to build a knowledge base for question analysis, which directly converts the input questions into SPARQL queries. So we have modified the conclusion and future work section to include this.*

The most important argument for the method presented in this section seems to be development time. Can you also say something about expressive power. For instance, does the formalism and method allow implementation of rules for comparative questions, list questions, complex (indirect) questions, etc, as used in the QALD competitions?

We thank for your question. We have added a new subsection 4.3.3 to illustrate the process of building knowledge base to handle the DBpedia and biomedical test questions in the QALD workshop. The illustration in section 4.3.3 specifies on comparative and list questions.

We have updated table 5 to detail the number of question structure patterns corresponding to each question structure type. Please find more complex questions in our online demo for question analysis at http://150.65.242.39:8080/KbEnQA/

5. Evaluation

You evaluate on an in house ontology, developed manually using Protege. This makes it very hard to compare your results to other work. Also, it seems the scope of the ontology is very limited, compared to current work on QA for DbPedia. The evaluation would be much more convincing, if it also included results for QA over an open domain LOD set such as DbPedia or similar, large, open, resource.

We thank for your suggestion. We have modified section 5.1.2 to include question analysis results on 50 DBpedia test questions from QALD-1 and 25 biomedical test questions from QALD-4. We also have modified the conclusion and future work section to include a future extension of our Vietnamese question answering system to an open QA domain.

(In this manuscript, we aim to present a language-independent question analysis approach and a Vietnamese ontology-based question answering system. So we separately evaluated those.)

Half of the correct answers in table 9 require interaction with users. Please explain what this amounted to.

Table 9 presents answering results to questions which the question analysis component successfully processed.

Half of the correct answers require interaction with users because the answer retrieval component asked the help from users to handle ambiguity cases, as illustrated in the first example in section 3.4. We have modified section 5.2 to clarify this.

RESPONSE TO REVIEWER 2:

1. I wonder that the necessary of developing a Vietnamese question analysis model for QA, Is there any prominent characters among Vietnamese and other language such as English and Chinese?

Following https://en.wikipedia.org/wiki/List_of_countries_by_number_of_Internet_users, Vietnam is in top-15 countries in the world, ranked by number of Internet users (about 43.90% of Vietnam's population). So it is necessary to develop Vietnamese information retrieval systems such as search engines or QA systems.

Vietnamese language has its own characteristics. For example, it uses the Latin script with nine accent marks, where a word can contain more than one token, thus leading to a difficult task of word segmentation. Therefore, existing information retrieval systems in other languages cannot work well in Vietnamese.

2. Most question answering systems mainly address the ambiguities in the question analysis step and the answer retrieval step. Is there any ambiguities in answering questions in this paper? What methods do you use to address this problem?

> *We thank for these questions. We have improved sections 4 and 3.4 to highlight the methods of addressing the ambiguities in question analysis and answer retrieval, respectively.*
>
> *The method to handle the ambiguities in question analysis is our knowledge acquisition approach which is presented in section 4 (We have restructured section 4.3 to clarify this).*
>
> *The method to handle the ambiguities in answer retrieval is to interact with users, as illustrated in the first example in section 3.4.*

3. This paper is rules based system, for example, the concepts and entities are determined using a manual dictionary. I wonder that it hard for scale other domain and language. Do you consider an extensible methods?

> *As explained to reviewer 1 (please see our response to #3.3.2), it is important to identify noun phrases rather than entities and concepts. So it is not hard to adapt our approach to other language (see sections 4.3.1 and 4.3.2) and domain (see new section 4.3.3). For example, we illustrated in section 4.3 the process of building a knowledge base for English question analysis without having any entity or concept information available.*
>
> *Because of the reason above, we did not consider to an extensible method. However, we have included a discussion on an extensible method for this issue, where the dictionary can be automatically constructed by extracting concepts from the target domain and their synonyms from available semantic lexicons like WordNet. Please see more details in the last paragraph of section 5.1.2.*

4. The related work lack some statistical semantic parsing methods such as works of Raymond J. Mooney and Percy Liang.

> *We thank the reviewer for pointing out related work. We added references [5,6,16,52].*

5. How to distinguish the questions contain multiply answers and just one answer. For example, the question “Which university does Pham Duc Dang study in and who tutors him?” contains two types of answers (university and person), and the question “List all students studying in K50 computer science course, who have hometown in Hanoi?” contains one type of answers even it cover two clauses.

> *As presented in sections 3.1 and 3.2, the first question “Which university does Pham Duc Dang study in and who tutors him?” has the **“Or”** question-structure type with two query-tuples corresponding to its two sub-questions. Meanwhile, the second question “List all students studying in K50 computer science course, who have hometown in Hanoi?” has the **“And”** question-structure type with two query-tuples corresponding to its two sub-questions.*

> *The difference between the **“And”** type and the **“Or”** type is: the **“And”** type returns the final answer as an intersection (i.e. overlap) of the answers for the sub-questions, while the **“Or”** type returns the final answer as an union of the answers for the sub-questions.*
>
> *We thank for the question. We have modified the appendix A to clarify the definitions of question-structure types.*

RESPONSE TO REVIEWER 3:

(1) Originality

Their claim that this is the such first system in Vietnamese is, as far as I know, valid. There is previous work on the system which is properly referenced, however it is not quite clear which part of the system was established in previous work and which part is new.

> *The key innovation of the current KbQAS version proposes a knowledge acquisition approach to systematically build a knowledge base for analyzing natural language questions. So, compared to the previous KbQAS version [36], the question analysis part is new while the answer retrieval part was established in [36].*
>
> *To clarify this, we have modified the introduction section, the last paragraph in section 3.3.3, the first paragraph in section 3.4 and the second paragraph in section 4.*

(2) Significance of the Results

The "Ripple Down Rules" are shown to significantly improve the performance of the rules which along with the drastic reported time savings and the high accuracy scores leads to a high significance of the results (the times used could be included in the table however, as it is a bit unclear what took how long exactly reading the description).

> *We have included the times in table 2 in section 4.1 as suggested.*

The knowledge base size of 78 instances, 15 concepts and 17 relations is too small for a realistic evaluation (also, instances are not part of the ontology as is mentioned), as it hides ambiguity, which is one of the main challenges faced by question answering approaches. Tthe test data does not have to be all of DBpedia but a few thousands of triples would already allow a much more realistic evaluation, especially as the approach is claimed to be applicable to other domains and languages. With a bigger dataset, a discussion of the complexity of the algorithm/scalability of the system and time measurements would be welcome additions. This is the major weak point in my opinion.

> *We have added new section 4.3.3 to illustrate the process of analyzing DBpedia and biomedical test questions, and modified section 5.1.2 to include question analysis results on these questions. The reviewer could also try our online demo for question analysis at http://150.65.242.39:8080/KbEnQA/*

We have also modified the conclusion and future work section to include a possible extension of our current Vietnamese ontology-based question answering system to be an open domain question answering system over linked open data.

(3) Quality of Writing

The writing style is certainly unusual but mostly in a refreshing way, with sharp observations that sometimes border on the comical, without feeling out of place in scientific writing. For example, instead of carefully defining web of document search and Question Answering and then analysing the difference, they go directly to the point: "Most current search engines take an (sic) user's query and returns (sic) a ranked list of related documents that are then scanned by the user to get the desired information. In contrast, the goal of QA systems is to give answers to the users' questions without involving the scanning process."

We thank for this comment. We have improved the introduction section as suggested.

As the above sentence shows, there are unfortunately also many basic spelling and grammar mistakes.

We have improved the paper to correct the mistakes.

Additionally, other parts are unnecessarily verbose. For example, they abbreviate "knowledge-based QA system for Vietnamese (KbQAS)" which I feel is a bit unwieldy in contrast to something simple like KS or even KQS. Also, they refer to it as the "KbQAS system", which is redundant, like "HIV virus" or "ATM machine". The abbreviation should also come directly after the term itself (I guess Vietnamese does not go into the abbreviation as the letter V is not appended).

Thanks for your suggestion. We have modified the paper to avoid the redundancy.

The reason, why the letter V is not appended, is that KbQAS could also be used to implicitly refer to our `language-independent` ***k**nowledge-**b**ased **q**uestion **a**nalysis approach. To avoid this confusion, we have modified the abstract and introduction sections.*

Some terms have a slightly different meaning than the one used in the paper. For example, the first stage of the pipeline is called "front-end" and the second stage the "back-end", although those terms signify the presentation and data access layer of an application.

We thank for pointing this out. We have changed the introduction section to remove the mentioned terms.

Some sections could be shortened a bit, such as 2.1 open-domain question answering. I do not think it is necessary to state the (undefined) performance percentage score of a system at TREC 2002, which was quite a while ago.

We agree with the reviewer. We have shortened the related work section.

2 Revised submission

Starts on next page.

Ripple Down Rules for Question Answering

Dat Quoc Nguyen [a], Dai Quoc Nguyen [b] and Son Bao Pham [c]
[a] *Department of Computing, Macquarie University, Australia*
E-mail: dat.nguyen@students.mq.edu.au
[b] *Department of Computational Linguistics, Saarland University, Germany*
E-mail: daiquocn@coli.uni-saarland.de
[c] *Faculty of Information Technology, VNU University of Engineering and Technology, Vietnam*
E-mail: sonpb@vnu.edu.vn

Abstract. Recent years have witnessed a new trend of building ontology-based question answering systems. These systems use semantic web information to produce more precise answers to users' queries. However, these systems are mostly designed for English. In this paper, we introduce an ontology-based question answering system named KbQAS which, to the best of our knowledge, is the first one made for Vietnamese. KbQAS employs our question analysis approach that systematically constructs a knowledge base of grammar rules to convert each input question into an intermediate representation element. KbQAS then takes the intermediate representation element with respect to a target ontology and applies concept-matching techniques to return an answer. On a wide range of Vietnamese questions, experimental results show that the performance of KbQAS is promising with accuracies of 84.1% and 82.4% for analyzing input questions and retrieving output answers, respectively. Furthermore, our question analysis approach can easily be applied to new domains and new languages, thus saving time and human effort.

Keywords: Question answering, Question analysis, Ripple Down Rules, Knowledge acquisition, Ontology, Vietnamese

1. Introduction

Accessing online resources often requires the support from the advanced information retrieval technologies to produce expected information. This brings new challenges to the construction of information retrieval systems such as search engines and question answering (QA) systems. Given an input query expressed in a keyword-based mechanism, most search engines return a long list of title and short snippet pairs ranked by their relevance to the input query. Then user is forced to scan the list to get the expected information, so this is a time consuming task [65]. Unlike the search engines, the QA systems directly produce an exact answer to an input question. In addition, the QA systems allow to specify the input question in natural language rather than in the keyword-based mechanism.

In general, an open-domain QA system aims to potentially answer any user's question. In contrast, a restricted-domain QA system only handles the questions related to a specific domain. Specifically, the traditional restricted-domain QA systems make use of the relational databases to represent target domains. Subsequently, with the advantages of semantic web, the recent restricted-domain QA systems employ knowledge bases such as ontologies as the target domains [30]. Thus, semantic markups can be used to add meta-information to return precise answers for complex natural language questions. This is an avenue which has not been actively explored for Vietnamese.

In this paper, we introduce the first ontology-based QA system for Vietnamese, which we call KbQAS. KbQAS consists of question analysis and answer retrieval components. The question analysis component uses a knowledge base of grammar rules for analyzing input questions; and the answer retrieval component is responsible for making the senses of the input questions with respect to a target ontology. The association between the two components is an intermediate representation element which is to capture the semantic structure of any input question. This intermediate element contains properties of the input question including question structure, question category, keywords and semantic constraints between the keywords.

The *key innovation* of KbQAS proposes a knowledge acquisition approach to systematically build a knowledge base for analyzing natural language questions. To convert a natural language question into an explicit representation in the QA systems, most previous works so far have used rule-based approaches to the best of our knowledge. The manual creation of rules in an ad-hoc manner is more expensive in terms of time, effort and error-prone because of the representation complexity and the variety of structure types of the questions. For example, rule-based methods, such as for English [26] and for Vietnamese as described in the first KbQAS version [35], manually define a list of pattern structures to analyze the questions. As rules are created in an ad-hoc manner, these methods share common difficulties in controlling the interaction between the rules and keeping the consistency among them. In our question analysis approach, however, we apply Single Classification Ripple Down Rules (SCRDR) knowledge acquisition methodology [10,46] to acquire the rules in a systematic manner, where the consistency between rules is maintained and the unintended interaction among rules is avoided. Our approach allows an easy adaptation to a new domain and a new language and saves time and effort of human experts.

The paper is organized as follows: we provide the related work in section 2. We describe KbQAS and our knowledge acquisition approach for question analysis in section 3 and section 4, respectively. We evaluate KbQAS in section 5. The conclusion will be presented in section 6.

2. Short overview of question answering

2.1. Open-domain question answering

The goal of an open-domain QA system is to automatically return an answer for every natural language question [21,62,31]. For example, such systems as START [23], FAQFinder [8] and AnswerBus [67] answer questions over the Web.Subsequently, the question-paraphrase recognition task is considered as one of the important tasks in QA. Many proposed approaches for this task are based on machine learning as well as knowledge representation and reasoning [7,22,47,66,16,5].

Since aroused by the QA track of the Text Retrieval Conference [58] and the multilingual QA track of the CLEF conference [41], many open-domain QA systems from the information retrieval perspective [24] have been introduced. For example, in the TREC-9 QA competition [57], the Falcon system [20] achieved the highest results. The innovation of Falcon focused on proposing a method using the WordNet [17] to boost its knowledge base. In the QA track of the TREC-2002 conference [59], the PowerAnswer system [33] was the most powerful system, using a deep linguistic analysis.

2.2. Traditional restricted-domain question answering

Usually linked to relational databases, the traditional restricted-domain QA systems are called natural language interfaces to databases. A natural language interface to a database (NLIDB) is a system that allows the users to access information stored in a database by typing questions using natural language expressions [2]. In general, NLIDB systems focus on converting the input question into an expression in the corresponding database query language. For example, the LUNAR system [63] transfers the input question into a parsed tree, and the tree is then directly converted into an expression in a database query language. However, it is difficult to create converting rules that directly transform the tree into the query expression.

Later NLIDBs, such as Planes [60], Eufid [50], PRECISE [45], C-Phrase [32] and the systems presented in [49,34], use semantic grammars to analyze questions. The semantic grammars consist of the hardwired knowledge orienting a specific domain, so these NLIDB systems need to develop new grammars whenever porting to a new knowledge domain.

Furthermore, some systems, such as TEAM [29] and Masque/sql [1], use the syntactic-semantic interpretation rules driving logical forms to process the input question. These systems firstly transform the input question into an intermediate logical expression of high level world concepts without any relation to the database structure. The logical expression is then converted to an expression in the database query language. Here, using the logical forms enables those systems to adapt to other domains as well as to different query languages [48]. In addition, there are many systems also using logical forms to process the input question as in [51,33,55,18,15,25,6].

2.3. Ontology-based question answering

As a knowledge representation of a set of concepts and their relations in a specific domain, an ontology can provide semantic information to handle the ambiguities, to interpret and answer the user questions in terms of QA [27]. The discussion on the construction approach of an ontology-based QA system can be found in [4]. This approach was then applied to build the MOSES system [3], with the focus on the question analysis. The following systems are some typical ontology-based QA systems.

The AquaLog system [26] performs semantic and syntactic analysis of the input question in the use of processing resources, including word segmentation, sentence segment and part-of-speech tagging, provided by the GATE framework [11]. When a question is asked, AquaLog transfers the question into a Query-Triple form of (*generic term, relation, second term*) containing the keyword concepts and relations in the question, using JAPE grammars in GATE. AquaLog then matches each element in the Query-Triple to an element in the target ontology to create an Onto-Triple, using string-based comparison methods and WordNet [17]. Evolved from AquaLog, the PowerAqua system [28] directs to open-domains by combining the knowledge from various heterogeneous ontologies which were autonomously created on the Semantic web. Meanwhile, the PANTO system [61] relies on the statistical Stanford parser to map an input question into a query-triple; the query-triple is then translated into an Onto-triple with the help of a lexicon of all entities from a given target ontology enlarged with WordNet synonyms; finally, the Onto-triple and potential words derived from the parse tree are used to produce a SPARQL query on the target ontology.

Using the gazetteers in the GATE framework, the QuestIO system [12] identifies the keyword concepts in an input question. Then QuestIO retrieves potential relations between the concepts before ranking these relations based on the similarity, distance and specificity scores; and so QuestIO creates formal SeRQL or SPARQL queries based on the concepts and the ranked relations. Later the FREyA system [13], the successor to QuestIO, allows users to enter questions in any form and interacts with the users to handle the ambiguities if necessary.

In the ORAKEL system [9], wh-questions are converted to F-Logic or SPARQL queries by using domain-specific Logical Description Grammars. Although ORAKEL supports compositional semantic constructions and obtains a promising performance, it involves a customization process of the domain-specific lexicon. Also, another interesting work over linked data as detailed in [54] proposed an approach to convert the syntactic-semantic representations of the input questions into the SPARQL templates. Furthermore, the Pythia system [53] relies on the ontology-based grammars generated from the Lexicalized Tree Adjoining Grammar tree to process complex questions. However, Pythia requires a manually created lexicon.

2.4. Question answering and question analysis for Vietnamese

Turning to the Vietnamese question answering, Nguyen and Le [34] introduced a Vietnamese NLIDB system using semantic grammars. Their system includes two main modules of the Query Translator (QTRAN) and the Text Generator (TGEN). QTRAN maps an input natural language question to an SQL query while TGEN generates an answer based on the table result of the SQL query. The QTRAN module uses the limited context-free grammars to convert the input question into a syntax tree via CYK algorithm [64]. The syntax tree is then converted into an SQL query by using a dictionary to identify names of attributes in database and names of individuals stored in these attributes. The TGEN module combines pattern-based and keyword-based approaches to make sense of the meta-data and relations in database tables to produce the answer.

In our first KbQAS conference publication [35], we reported a hard-wire approach to convert input questions into intermediate representation elements which are then used to extract the corresponding elements from a target ontology to return answers. Later, Phan and Nguyen [44] described a method to map Vietnamese questions into triple-like formats of (*Subject*, *Verb*, *Object*). Subsequently, Nguyen and Nguyen [39] presented another ontology-based QA system for Vietnamese, where keywords in an input question are identified by using pre-defined templates, and these keywords are then used to produce a SPARQL query to retrieve a triple-based answer from a target ontology. In addition, Tran et al. [52] described the VPQA system to answer person name-related questions while Nguyen et al. [40] presented another NLIDB system to answer questions in the economic survey domain.

3. Our KbQAS question answering system

This section is to describe the overview of KbQAS. The architecture of KbQAS, as shown in Figure 1, contains two components of the Natural language question analysis engine and the Answer retrieval.

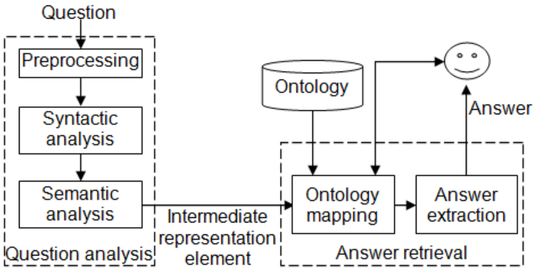


Figure 1. System architecture of KbQAS.

The question analysis component consists of three modules: preprocessing, syntactic analysis and semantic analysis. This component takes the user question as an input and returns an intermediate element representing the input question in a compact form. The role of the intermediate representation element is to provide the structured information about the input question for later process of answer retrieval.

The answer retrieval component contains two modules of Ontology mapping and Answer extraction. It takes the intermediate representation element produced by the question analysis component and an Ontology as its input to generate the answer.

3.1. Intermediate representation of an input question

Unlike AquaLog [26], the intermediate representation element in KbQAS covers a wider variety of question types. This element consists of a question-structure and one or more query-tuples in the following format:

(sub-structure, question-category, $Term_1$, $Relation$, $Term_2$, $Term_3$*)*

where $Term_1$ represents a concept (i.e. an object class), excluding the cases of "*Affirm*", "*Affirm_3Term*" and "*Affirm_MoreTuples*" question-structures. In addition, $Term_2$ and $Term_3$ represent entities (i.e. objects or instances), excluding the cases of "*Definition*" and "*Compare*" question-structures. The $Relation$ (i.e. property) is a semantic constraint between the terms.

We define the following question-structures: *Normal, UnknTerm, UnknRel, Definition, Compare, ThreeTerm, Clause, Combine, And, Or, Affirm_MoreTuples, Affirm, Affirm_3Term*, and question categories: *What, When, Where, Who, HowWhy, YesNo, Many, ManyClass, List* and *Entity*. See the Appendixes **A** and **B** for details of these definitions.

A simple question has only one query-tuple and its question-structure is the sub-structure in the query-tuple. A complex question, such as a composite one, has several sub-questions, where each sub-question is represented by a separate query-tuple, and the question-structure captures this composite factor. For example, the question:

"Phạm Đức Đăng học trường đại học nào và được hướng dẫn bởi ai?"

Which university does Pham Duc Dang study in and who tutors him?

has the "*Or*" question-structure and two query-tuples where **?** represents a missing attribute: *(Normal, Entity, trường đại học*$_{university}$*, học*$_{study}$*, Phạm Đức Đăng*$_{Pham\ Duc\ Dang}$*, ?)* and *(UnknTerm, Who, ?, hướng dẫn*$_{tutor}$*, Phạm Đức Đăng*$_{Pham\ Duc\ Dang}$*, ?)*.

The intermediate representation element is designed so that it can represent various types of question-structures. Therefore, attributes such as $Relation$ or terms in the query-tuple can be missing. For example, a question has the "*Normal*" question-structure if it has only one query-tuple and $Term_3$ is missing.

3.2. An illustrative example

For demonstration[1] [38] and evaluation purposes, we reuse an ontology which models the organizational system of the VNU University of Engineering and Technology, Vietnam. The ontology contains 15 concepts such as "trường$_{school}$", "giảng viên$_{lecturer}$" and "sinh viên$_{student}$", 17 relations or properties such as "học$_{study}$", "giảng dạy$_{teach}$" and "là sinh viên của$_{is\ student\ of}$", and 78 instances, as described in our first KbQAS version [35].

Given a complex-structure question:

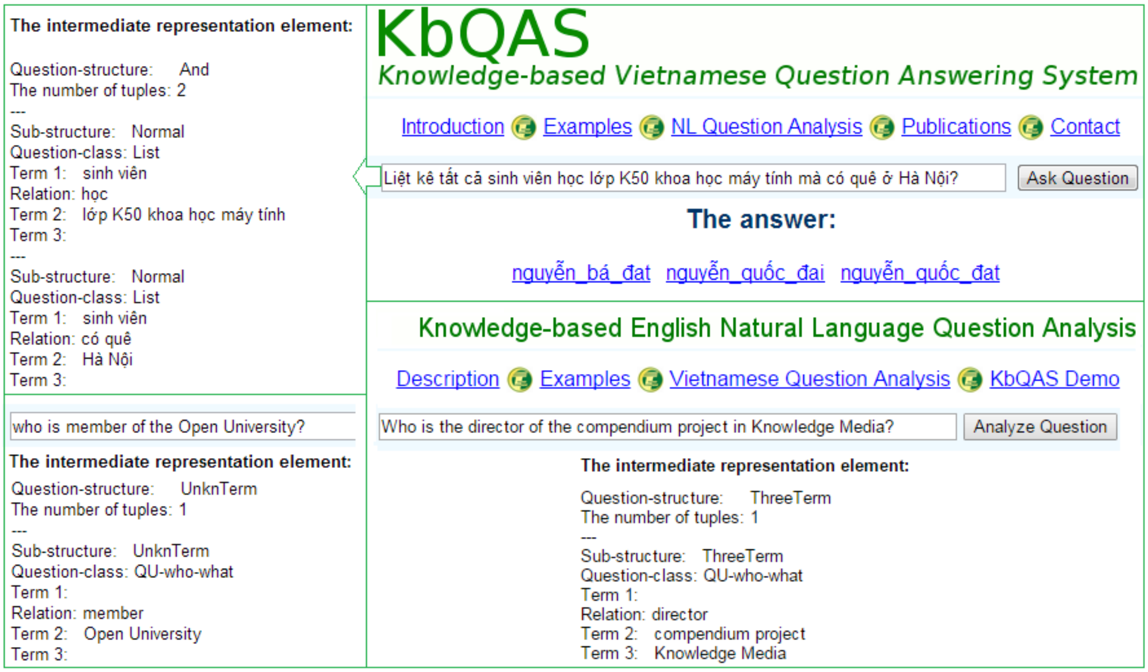
"Liệt kê tất cả sinh viên học lớp K50 khoa học máy tính mà có quê ở Hà Nội?"

"List all students studying in K50 computer science course, who have hometown in Hanoi?"

The question analysis component determines that this question has the "*And*" question-structure with two query-tuples *(Normal, List, sinh viên*$_{student}$*, học*$_{study}$*, lớp K50 khoa học máy tính*$_{K50\ computer\ science\ course}$*, ?)* and *(Normal, List, sinh viên*$_{student}$*, có quê*$_{has\ hometown}$*, Hà Nội*$_{Hanoi}$*, ?)*.

In the Answer retrieval component , the Ontology mapping module maps the query-tuples to ontology-tuples: *(sinh viên*$_{student}$ *, học*$_{study}$ *, lớp K50 khoa học máy tính*$_{K50\ computer\ science\ course}$*)* and *(sinh viên*$_{student}$ *, có quê*$_{has\ hometown}$ *, Hà Nội*$_{Hanoi}$*)*.

[1] The KbQAS is available at http://150.65.242.39:8080/KbQAS/ with an intro video on YouTube at http://youtu.be/M1PHvJvv1Z8


KbQAS
Knowledge-based Vietnamese Question Answering System
Introduction
Examples
NL Question Analysis
Publications
Contact
Ask Question
Knowledge-based English Natural Language Question Analysis
Description
Examples
Vietnamese Question Analysis
KbQAS Demo
Analyze Question

The intermediate representation element:

Question-structure: And
The number of tuples: 2

Sub-structure: Normal
Question-class: List
Term 1: sinh viên
Relation: học
Term 2: lớp K50 khoa học máy tính
Term 3:

Sub-structure: Normal
Question-class: List
Term 1: sinh viên
Relation: có quê
Term 2: Hà Nội
Term 3:

Liệt kê tất cả sinh viên học lớp K50 khoa học máy tính mà có quê ở Hà Nội?

The answer:

nguyễn_bá_đat nguyễn_quốc_đai nguyễn_quốc_đat

who is member of the Open University?

The intermediate representation element:

Question-structure: UnknTerm
The number of tuples: 1

Sub-structure: UnknTerm
Question-class: QU-who-what
Term 1:
Relation: member
Term 2: Open University
Term 3:

Who is the director of the compendium project in Knowledge Media?

The intermediate representation element:

Question-structure: ThreeTerm
The number of tuples: 1

Sub-structure: ThreeTerm
Question-class: QU-who-what
Term 1:
Relation: director
Term 2: compendium project
Term 3: Knowledge Media

Figure 2. Illustrations of question analysis and question answering.

For each ontology-tuple, the answer extraction module finds all satisfied instances in the target ontology, and it then generates an answer based on the “*And*” question-structure and the “*List*” question-category. Figure 8 shows the answer.

3.3. Natural language question analysis component

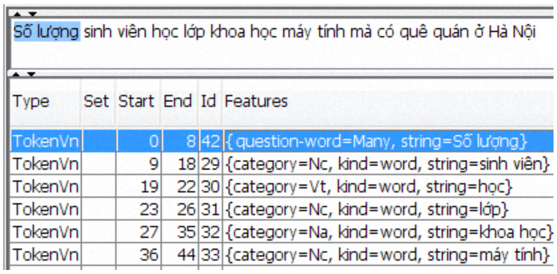
Natural language question analysis component is the first component in any QA system. When a question is asked, the task of this component is to convert the input question into an intermediate representation which is then used in the rest of the system.

KbQAS makes the use of the JAPE grammars in the GATE framework [11] to specify semantic annotation-based regular expression patterns for question analysis, in which existing linguistic processing modules for Vietnamese including Word segmentation and Part-of-speech tagging [42] are wrapped as GATE plug-ins. The results of the wrapped plug-ins are annotations covering sentences and segmented words. Each annotation has a set of feature-value pairs. For example, a word has a *“category”* feature storing its part-of-speech tag. This information can then be reused for further processing in subsequent modules. The new question analysis modules of preprocessing, syntactic analysis and semantic analysis in KbQAS are specifically designed to handle Vietnamese questions using patterns over existing linguistic annotations.

3.3.1. Preprocessing module

The preprocessing module generates *TokenVn* annotations representing a Vietnamese word with features, such as part-of-speech, as displayed in Figure 3. Vietnamese is a monosyllabic language; hence, a word can contain more than one token. So there are words or word-phrases which are indicative of the question-categories, such as *“phải không*$_{is\ that|are\ there}$*”*, *“là bao nhiêu*$_{how\ many}$*”*, *“ở đâu*$_{where}$*”*, *“khi nào*$_{when}$*”* and *“là cái gì*$_{what}$*”*. However, the Vietnamese word segmentation module was not trained for question domain. In this module, therefore, we identify those words or phrases, and label them as single *TokenVn* annotations with the *“question-word”* feature and its semantic category like $HowWhy_{cause\ |\ method}$, $YesNo_{true\ or\ false}$, $What_{something}$, $When_{time\ |\ date}$, $Where_{location}$, $Many_{number}$ or Who_{person}. In fact, this information will be used to create rules in the syntactic analysis module at a later stage.

We also label special-words, such as abbreviations of words on a special-domain, and phrases that refer to a comparison, such as *“lớn hơn*$_{greater\ than}$*”*, *“nhỏ hơn hoặc bằng*$_{less\ than\ or\ equal\ to}$*”* and the like, by single *TokenVn* annotations.



Số lượng sinh viên học lớp khoa học máy tính mà có quê quán ở Hà Nội

Type	Set	Start	End	Id	Features
TokenVn		0	8	42	{question-word=Many, string=Số lượng}
TokenVn		9	18	29	{category=Nc, kind=word, string=sinh viên}
TokenVn		19	22	30	{category=Vt, kind=word, string=học}
TokenVn		23	26	31	{category=Nc, kind=word, string=lớp}
TokenVn		27	35	32	{category=Na, kind=word, string=khoa học}
TokenVn		36	44	33	{category=Nc, kind=word, string=máy tính}

Figure 3. Examples of TokenVn annotations.

3.3.2. *Syntactic analysis*

The syntactic analysis module is responsible for identifying concepts, entities and the relations between them in the input question. This module uses the *TokenVn* annotations which are the output of the preprocessing module.

Table 1

JAPE grammar for identifying Vietnamese noun phrases

({TokenVn.category == “Pn”})**?**	Quantity pronoun
({TokenVn.category == “Nu”} |	Concrete noun
{TokenVn.category == “Nn”})**?**	Numeral noun
({TokenVn.string == “cái”} |	“cái$_{the}$”
{TokenVn.string == “chiếc”})**?**	“chiếc$_{the}$”
({TokenVn.category == “Nt”})**?**	Classifier noun
({TokenVn.category == “Nc”} |	Countable noun
{TokenVn.category == “Ng”} |	Collective noun
{TokenVn.category == “Nu”} |	
{TokenVn.category == “Na”} |	Abstract noun
{TokenVn.category == “Np”})**+**	Proper noun
({TokenVn.category == “Aa”} |	Quality adjective
{TokenVn.category == “An”})**?**	Quantity adjective
({TokenVn.string == “này”} |	“này$_{this;\ these}$”
{TokenVn.string == “kia”} |	“kia$_{that;\ those}$”
{TokenVn.string == “ấy”} |	“ấy$_{that;\ those}$”
{TokenVn.string == “đó”})**?**	“đó$_{that;\ those}$”

Concepts and entities are normally expressed in noun phrases. Therefore, it is crucial to identify noun phrases in order to generate the query-tuple. Based on the Vietnamese language grammar [14], we use the JAPE grammars to specify patterns over annotations as shown in Table 1. When a noun phrase is matched, a *NounPhrase* annotation is created to mark up the noun phrase. In addition, a *“type”* feature of the *NounPhrase* annotation is used to determine whether concept or entity is covered by the noun phrase, using the following heuristics: if the noun phrase contains a single noun (not including numeral nouns) and does not contain a proper noun, it covers a concept. If the noun phrase contains a proper noun or at least three single nouns, it covers an entity. Otherwise, the “*type*” feature value is determined by using a dictionary.[2]

Furthermore, the question-phrases are detected by using the matched noun phrases and the question-words which are identified by the preprocessing module. *QuestionPhrase* annotations are generated to cover the question-phrases, with a *“category”* feature that gives the information about question categories.

The next step is to identify relations between noun phrases or between a noun phrase and a question-phrase. When a phrase is matched by one of the relation patterns, a *Relation* annotation is created to markup the relation. We use four grammar patterns to determine relation phrases as following:

(Verb)**+** (Noun Phrase$_{type==Concept}$) (Preposition)(Verb)**?**
(Verb)**+**((Preposition)(Verb)**?**)**?**
((“có$_{have\|has}$”)|(Verb))**+** (Adjective) (Preposition) (Verb)**?**
(“có$_{have\|has}$”) ((Noun Phrase$_{type==Concept}$)|(Adjective)) (“là$_{is\|are}$”)

For example, with the first question in Figure 4:

“liệt kê tất cả các sinh viên có quê quán ở Hà Nội?”

“list all students who have hometown in Hanoi?”

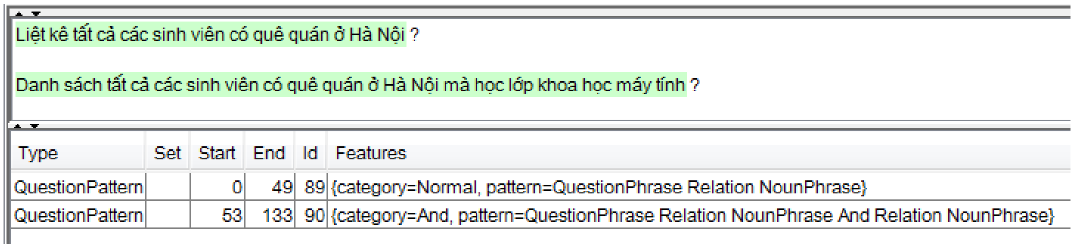
[QuestionPhrase: liệt kê$_{list}$ [NounPhrase: tất cả các sinh viên$_{all\ students}$]] [Relation: có quê quán ở$_{have\ hometown\ in}$] [NounPhrase: Hà Nội$_{Hanoi}$]

The phrase *“có quê quán ở$_{have\ hometown\ in}$”* is the relation linking the question-phrase *“liệt kê tất cả các sinh viên$_{list\ all\ students}$”* and the noun phrase *“Hà Nội$_{Hanoi}$”*.

3.3.3. *Semantic analysis module*

The semantic analysis module aims to identify the question-structure and produce the query-tuples *(sub-structure, question-category, $Term_1$, Relation, $Term_2$, $Term_3$)* as the intermediate representation element of the input question, using the *TokenVn*, *NounPhrase*, *Relation* and *QuestionPhrase* annotations re-

[2] The dictionary contains concepts which are extracted from the target ontology. However, there is no publicly available WordNet-like lexicon for Vietnamese. So we manually add synonyms of the extracted concepts to the dictionary.



Liệt kê tất cả các sinh viên có quê quán ở Hà Nội ?

Danh sách tất cả các sinh viên có quê quán ở Hà Nội mà học lớp khoa học máy tính ?

Type	Set	Start	End	Id	Features
QuestionPattern		0	49	89	{category=Normal, pattern=QuestionPhrase Relation NounPhrase}
QuestionPattern		53	133	90	{category=And, pattern=QuestionPhrase Relation NounPhrase And Relation NounPhrase}

Figure 4. Examples of question-structure patterns.

turned by the two previous modules. Existing *NounPhrase* annotations and *Relation* annotations are potential candidates for terms and relations in the query-tuples, respectively. In addition, *QuestionPhrase* annotations are used to detect the question-category.

In the first KbQAS version [35], following AquaLog [26], we developed an ad-hoc approach to detect structure patterns of questions and then use these patterns to generate the intermediate representation elements. For example, Figure 4 presents the detected structure patterns of two example questions *"Liệt kê tất cả các sinh viên có quê quán ở Hà Nội?"* ("List all students who have hometown in Hanoi?") and *"Danh sách tất cả các sinh viên có quê quán ở Hà Nội mà học lớp khoa học máy tính?"* ("List all students having hometown in Hanoi, who study in computer science course?"). We can describe these questions by using annotations generated by the preprocessing and syntactic analysis modules as following:

[QuestionPhrase: Liệt kê tất cả các sinh viên$_{List\ all\ students}$] [Relation: có quê quán ở$_{have\ hometown\ in}$] [NounPhrase: Hà Nội$_{Hanoi}$]

and

[QuestionPhrase: Liệt kê tất cả các sinh viên$_{List\ all\ students}$] [Relation: có quê quán ở$_{have\ hometown\ in}$] [NounPhrase: Hà Nội$_{Hanoi}$] [And: [TokenVn: mà$_{and}$]] [Relation: học$_{study\ in}$] [NounPhrase: lớp khoa học máy tính$_{computer\ science\ course}$]

The intermediate representation element of an input question is created in a hard-wire manner linking every detected structure pattern via JAPE grammars. This hard-wire manner takes a lot of time and effort to handle new patterns. For example in Figure 4, the hard-wire approach is unable to reuse the detected structure pattern of the first question to identify the structure pattern of the second question. Since JAPE grammar rules were created in an ad-hoc manner, the hard-wire approach encounters itself common difficulties in managing the interaction among rules and keeping consistency.

Consequently, in this module, we solve the mentioned difficulties by proposing a knowledge acquisition approach for semantic analysis of input questions, as detailed in the section 4. In this paper, this is considered as the key innovation of KbQAS.

3.4. Answer retrieval component

As presented in the first KbQAS version [35], the Answer retrieval component includes two modules of Ontology mapping and Answer extraction as shown in Figure 1. It takes the intermediate representation produced by the question analysis component and a target ontology as its input to generate an answer. To develop the Answer retrieval component in KbQAS, we employed the Relation similarity service component of AquaLog [26].

The task of the Ontology mapping module is to map terms and relations in the query-tuple to concepts, instances and relations in the target ontology by using string names. If an exact match is not possible, we use the string distance algorithm [56] and the dictionary containing concepts and their synonyms to find near-matched elements from the target ontology, with the similarity measure above a certain threshold.

In case of the ambiguity is still present, KbQAS interacts with users by showing different options, and the users then choose the suitable ontology element. For example, given the question *"liệt kê tất cả các sinh viên học lớp khoa học máy tính ?"* ("list all students studying in computer science course ?"), the question analysis component produce a query-tuple *(Normal, List, sinh viên$_{student}$, học$_{study}$, lớp khoa học máy tính$_{computer\ science\ course}$, ?)*. Because the Ontology mapping module cannot find the exact instance corresponding with *"lớp khoa học máy tính$_{computer\ science\ course}$"* in the target ontology, it requires the user to select between *"lớp K50 khoa học máy tính$_{K50\ computer\ science\ course}$"* - an instance of class *"lớp$_{course}$"* and *"bộ môn khoa học máy* *tính*$_{computer\ science\ department}$" - an instance of class "*bộ môn*$_{department}$".

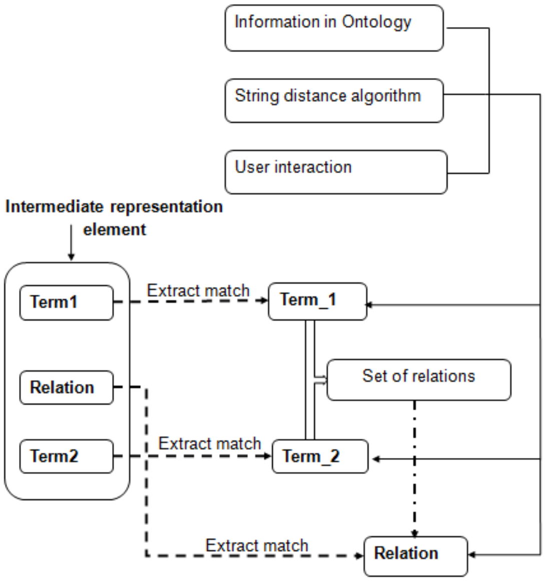


Figure 5. Ontology mapping module for the query-tuple with two terms and one relation.

Following AquaLog, for each query-tuple, the result of the Ontology mapping module is an ontology-tuple where the terms and relations in the query-tuple are now the corresponding elements from the target ontology. How the Ontology mapping module finds the corresponding elements from the target ontology depends on the question-structure. For example, when the query-tuple contains $Term_1$, $Term_2$ and $Relation$ with $Term_3$ missing, the mapping process follows the diagram shown in Figure 5. The mapping process first tries to match $Term_1$ and $Term_2$ with concepts or instances in the target ontology. Then the mapping process finds a set of potential relations between the two mapped concepts/instances from the target ontology. The ontology relation is finally identified by mapping $Relation$ to a relation in the potential relation set, using the similar manner of mapping a term to a concept or an instance.

With the ontology-tuple, the answer extraction module finds all individuals of the ontology concept corresponding to $Term_1$, having the ontology relation with the ontology individual corresponding to $Term_2$. The answer extraction module then returns the answer based on the question-structure and question-category. See the definitions of question-structure and question-category types in the appendixes **A** and **B**.

4. Single Classification Ripple Down Rules for Question Analysis

As mentioned in section 3.3.3, due to the representation complexity and the variety of question structures, manually creating grammar rules in an ad-hoc manner is very expensive and error-prone. For example, such rule-based approaches as presented in [26,35,44] manually defined a list of sequence pattern structures to analyze questions. Since rules were created in an ad-hoc manner, these approaches share common difficulties in managing the interaction between rules and keeping consistency among them.

This section is to introduce our knowledge acquisition approach[3] to analyze natural language questions by applying the SCRDR methodology [10,46] to acquire rules incrementally. *Our contribution* focuses on the semantic analysis module by proposing a JAPE-like rule language and a systematic processing to create rules in a manner that the interaction among rules is controlled and the consistency is maintained. Compared to the first KbQAS version [35], this is the key innovation of the current KbQAS version.

A SCRDR knowledge base is built to identify the question-structures and to produce the query-tuples as the intermediate representations of the input questions. We outline the SCRDR methodology and propose a rule language for extracting the intermediate representation of a given question in sections 4.1 and 4.2, respectively. We then illustrate the process of systematically constructing a SCRDR knowledge base for analyzing questions in section 4.3.

4.1. Single Classification Ripple Down Rules

This section presents the basic idea of Single Classification Ripple Down Rules (SCRDR) [10,46] which inspired our knowledge acquisition approach for question analysis. A SCRDR tree is a binary tree with two distinct types of edges. These edges are typically called *except* and *false* edges. Associated with each node in a tree is a *rule*. A rule has the form: *if* α *then* β where α is called the *condition* and β is called the *conclusion*.

Cases in SCRDR are evaluated by passing a case to the root node of the SCRDR tree. At any node in the SCRDR tree, if the condition of the rule at a node η is

[3] The English question analysis demonstration is available online at http://150.65.242.39:8080/KbEnQA/, and the Vietnamese question analysis demonstration is available online at http://150.65.242.39:8080/KbVnQA/.

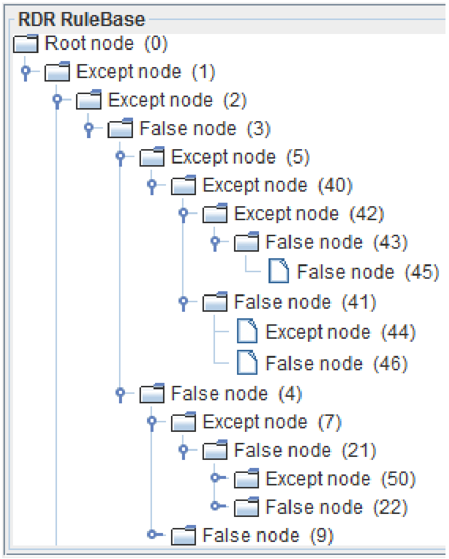


Figure 6. A part of the SCRDR tree for English question analysis.

satisfied by the case (so the node η *fires*), the case is passed on to the *except* child node of the node η using the *except* edge if it exists; otherwise, the case is passed on to the *false* child node of the node η. The conclusion given by this process is the conclusion from the node which *fired* last.

Given the question case “who are the partners involved in AKT project?” and the SCRDR tree in Figure 6, it is satisfied by the rule at the root node (0). Then it is passed to node (1) using the except edge. As the case satisfies the condition of the rule at node (1), it is passed to node (2) using the except edge. Because the case does not satisfy the condition of the rule at node (2), it is then passed to node (3) using the false edge. As the case satisfies the conditions of the rules at nodes (3), (5) and (40), it is passed to node (42), using except edges. Since the case does not satisfy the conditions of the rules at nodes (42), (43) and (45), we have the evaluation path (0)-(1)-(2)-(3)-(5)-(40)-(42)-(43)-(45) with the fired node (40). Given another case of “in which projects is enrico motta working on”, it satisfies the conditions of the rules at nodes (0), (1) and (2); as node (2) has no except child node, we have the evaluation path (0)-(1)-(2) and the fired node (2).

A new node containing a new exception rule is added to an SCRDR tree when the evaluation process returns an *incorrect* conclusion. The new exception node is attached to the last node in the evaluation path of the given case with *except* edge if the last node is the fired node; otherwise, it is attached with *false* edge.

To ensure that a conclusion is always given, the root node, called the *default* node, typically contains a trivial condition which is always satisfied. The rule at the default node, the default rule, is the unique rule which is not an exception rule of any other rule. For example, the default rule “*if* **True** *then* **null**” from the SCRDR tree in Figure 6 means that its *True* condition satisfies every question, however, its *null* conclusion produces an empty intermediate representation element for every question. Started with a SCRDR knowledge base consisting of only the default node, the process of building the knowledge base can be performed automatically [37] or manually [43,36].

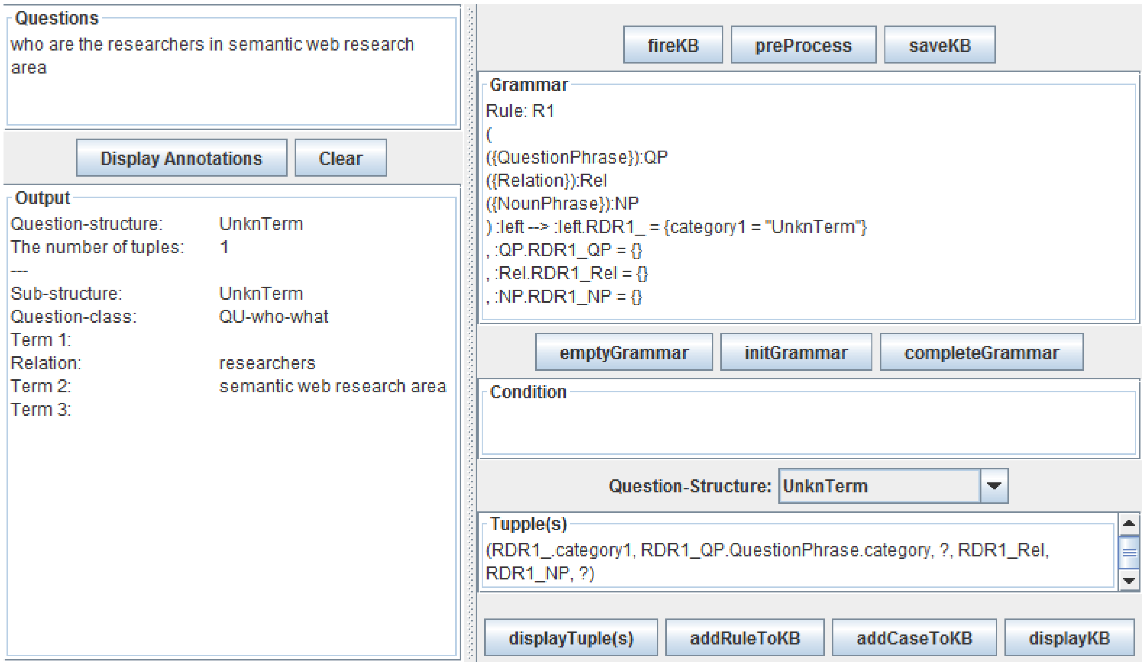
In the SCRDR tree from Figure 6, the rule at node (1) (simply, rule 1) is an exception rule of the default rule 0. Rule 2 is an exception rule of rule 1. As node (3) is the false-child node of node (2), the rule 3 is also an exception rule of rule 1. Furthermore, both rules 4 and 9 are also exception rules of rule 1. Similarly, all rules 40, 41 and 46 are exception rules of rule 5 while all rules 42, 43 and 45 are exception rules of rule 40. Therefore, the exception structure of the SCRDR tree extends to 5 levels, for examples: rules 1 at layer 1; rules 2, 3, 4 and 9 at layer 2; rules 5, 7, 21 and 22 at layer 3; and rules 40, 41, 46 and 50 at layer-4; and rules 42, 43, 44 and 45 at the layer-5 exception structure.

4.2. *Rule language*

A rule is composed of a condition part and a conclusion part. A condition is a regular expression pattern over annotations using JAPE grammar in GATE [11]. It can also post new annotations over matched phrases of the pattern’s sub-components. As annotations have feature-value pairs, we can impose constraints on the annotations in the pattern by specifying that a feature of an annotation must have a particular value. The following example shows the posting of an annotation over the matched phrase:

(
({TokenVn.string == “liệt kê$_{list}$”} | {TokenVn.string == “chỉ ra$_{show}$”})
{NounPhrase.type == “Concept”}
) :qp $\dashrightarrow$:qp.QuestionPhrase = {category = “List”}

Every complete pattern followed by a label must be enclosed by round brackets. In the above pattern, the label is qp. The pattern would match phrases starting with a *TokenVn* annotation covering either the word *“liệt kê$_{list}$”* or the word *“chỉ ra$_{show}$”*, followed by a *NounPhrase* annotation covering a *con-*



Questions

who are the researchers in semantic web research area

Output

```
Question-structure:      UnknTerm
The number of tuples:    1
---
Sub-structure:           UnknTerm
Question-class:          QU-who-what
Term 1:
Relation:                researchers
Term 2:                  semantic web research area
Term 3:
```

Grammar

```
Rule: R1
(
({QuestionPhrase}):QP
({Relation}):Rel
({NounPhrase}):NP
) :left --> :left.RDR1_ = {category1 = "UnknTerm"}
, :QP.RDR1_QP = {}
, :Rel.RDR1_Rel = {}
, :NP.RDR1_NP = {}
```

Question-Structure: UnknTerm

Tupple(s)

(RDR1_.category1, RDR1_QP.QuestionPhrase.category, ?, RDR1_Rel, RDR1_NP, ?)

Figure 7. The graphic user interface for knowledge base construction.

cept-typed noun phrase. When applying this pattern on a text fragment, *QuestionPhrase* annotations having the *"category"* feature with its *"List"* value would be posted over phrases matched by the pattern. Furthermore, the condition part of a rule can include additional constraints. See examples of the additional constraints from the constructions of rules (40) and (45) in section 4.3.

The conclusion part of a rule produces an intermediate representation containing the question-structure and the query-tuples, where each attribute in the query-tuples is specified by a newly posted annotation from matching the rule's condition, in the following order:

(sub-structure, question-category, $Term_1$*, Relation,* $Term_2$*,* $Term_3$*)*

All newly posted annotations have the same *"RDR"* prefix and the rule index so that a rule can refer to annotations of its parent rules. Examples of rules and how rules are created and stored in exception structure will be explained in details in section 4.3.

Given an input question, the condition of a rule is satisfied if the whole input question is matched by the condition pattern. The conclusion of the fired rule produces the intermediate representation element of the input question. To create rules for matching the structures of questions, we use patterns over annotations returned by the preprocessing and syntactic analysis modules.

4.3. Knowledge Acquisition Process

It is because that the main focus of our approach is on the process of creating the rule-base system, so it is language independent. The language-specific part is in the rules itself. So, in this section, we illustrate the process of building a SCRDR knowledge base to analyze English questions. Figure 7 shows the graphic user interface to construct SCRDR knowledge bases.

We reused the JAPE grammars which were developed to identify the noun phrases, question phrases and relation phrases in AquaLog [26]. Based on *Token* annotations which are generated as outputs of the English tokenizer, sentence splitter and POS tagger in the GATE framework [11], the JAPE grammars produce *NounPhrase*,[4] *QuestionPhrase* and *Relation* annotations, and other annotation kinds such as "*Noun*", "*Verb*" or "*Preps*" annotations for covering nouns, verbs or prepositions, respectively. We also reused question-category definitions from AquaLog.

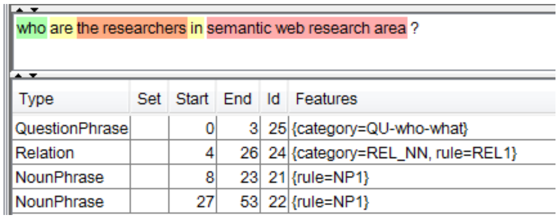
[4] Here annotations are generated without any concept or entity type information.

For illustrations in sections 4.3.1 and 4.3.2, we employed a training set of 170 English questions,[5] which AquaLog [26] analyzed successfully, to construct the SCRDR knowledge base in Figure 6. These questions are specified to the Knowledge Media Institute and its research area on Semantic web.

4.3.1. Reusing detected question-structures

In contrast to the example in section 3.3.3 with respect to Figure 4, we start with demonstrations of reusing detected question-structure patterns.

With the question *"who are the researchers in semantic web research area ?"*



who are the researchers in semantic web research area ?

Type	Set	Start	End	Id	Features
QuestionPhrase		0	3	25	{category=QU-who-what}
Relation		4	26	24	{category=REL_NN, rule=REL1}
NounPhrase		8	23	21	{rule=NP1}
NounPhrase		27	53	22	{rule=NP1}

Figure 8. Annotations in the graphic user interface of GATE [11].

[QuestionPhrase: who] [Relation: are the researchers in] [NounPhrase: semantic web research area]

Supposed we start with a knowledge base containing only the default rule **R0**. Given the question, **R0** is the fired rule that gives an incorrect conclusion of empty intermediate representation element. This can be corrected by adding the following rule **R1** as an exception rule of **R0**. In the knowledge base, node (1) containing **R1** is added as the except-child node of the default node, as shown in Figure 6.

Rule: R1
(
({QuestionPhrase}):qp
({Relation}):rel
({NounPhrase}):np
) :left --→ :left.RDR1_ = {category1 = "UnknTerm"}
, :qp.RDR1_QP = { }
, :rel.RDR1_Rel = { }
, :np.RDR1_NP = { }

Conclusion:
"*UnknTerm*" question-structure and one query-tuple (*RDR1_.category1, RDR1_QP.QuestionPhrase.category, ?, RDR1_Rel, RDR1_NP, ?*)

If the condition of **R1** matches the whole input question, a new *RDR1_* annotation will be created to entirely cover the input question. In addition, new annotations *RDR1_QP*, *RDR1_Rel* and *RDR1_NP* will be created to cover the same question phrase, relation phrase and noun phrase as the *QuestionPhrase*, *Relation* and *NounPhrase* annotations, respectively.

When node (1) fired, the input question has one query-tuple where the *sub-structure* attribute takes the value of the *"category1"* feature of the *RDR1_* annotation; the *question-category* attribute takes the value of the *"category"* feature of the *QuestionPhrase* annotation which is in the same span to the *RDR1_QP* annotation. In addition, the $Relation$ and $Term_2$ attributes take values of the strings covered by the *RDR1_Rel* and *RDR1_NP* annotations, respectively, while $Term_1$ and $Term_3$ are missing. The example of firing the question at node (1) is displayed in Figure 7.

Assumed that, in addition to **R0** and **R1**, the current knowledge base contains rule **R2** as an exception rule of **R1**, for which node (2) containing **R2** is the except-child node of node (1), as shown in Figure 6.

With the question *"which universities are Knowledge Media Institute collaborating with ?"*

[RDR1_: [RDR1_QP: which universities] [RDR1_Rel: are] [RDR1_NP: Knowledge Media Institute]] [Relation: collaborating with]

We have the evaluation path of (0)-(1)-(2) with the fired node (1). However, **R1** produces an incorrect conclusion of the "*UnknTerm*" question-structure and one query-tuple *(UnknTerm, QU-whichClass, ?, ?, Knowledge Media Institute, ?)*. It is because the *RDR1_* annotation only covers a part of the question and *"are"* is not considered as a relation. The following rule **R3** is added as an exception rule of **R1**:

Rule: R3
(
{RDR1_} ({Relation}):rel
) :left --→ :left.RDR3_ = {category1 = "Normal"}
, :rel.RDR3_Rel = { }

Conclusion:
"*Normal*" question-structure and one query-tuple (*RDR3_.category1, RDR1_QP.QuestionPhrase.category, RDR1_QP, RDR3_Rel, RDR1_NP, ?*)

In the knowledge base, node (3) containing **R3** is appended as the false-child node of node (2) which is the last node in the evaluation path. Regarding to the input question "*which universities are Knowledge Media Institute collaborating with ?*", we have a new evaluation path of (0)-(1)-(2)-(3) with the fired node (3). So **R3** produces a correct intermediate represen-

[5] http://technologies.kmi.open.ac.uk/aqualog/examples.html

tation element of the question, consisting of the "*Normal*" question-structure and one query-tuple *(Normal, QU-whichClass, universities, collaborating, Knowledge Media Institute, ?)*.

Subsequently, another question makes an addition of rule **R4** which is also an exception rule of **R1**. In the knowledge base, the node (4) containing **R4** is appended as the false-child node of node (3).

With the question *"who are the partners involved in AKT project?"*

[RDR3_: [RDR1_QP: who] [RDR1_Rel: are] [RDR1_NP: the partners] [RDR3_Rel: involved in]] [NounPhrase: AKT project]

We have the evaluation path (0)-(1)-(2)-(3) and node (3) is the fired node. But **R3** returns a wrong conclusion as the *RDR3_* annotation covers a part of the question. The following rule **R5** is added as an exception rule of **R3** to correct the returned conclusion:

Rule: R5
(
{RDR3_} ({NounPhrase}):np
) :left --→ :left.RDR5_ = {category1 = "Normal"}
, :np.RDR5_NP = { }

Conclusion:
"*Normal*" question-structure and one query-tuple
(*RDR5_.category1, RDR1_QP.QuestionPhrase.category, RDR1_NP, RDR3_Rel, RDR5_NP, ?*)

As node (3) is the last node in the evaluation path, node (5) containing **R5** is attached as the except-child node of node (3). Using **R5**, we have a correct conclusion consisting of the "*Normal*" question-structure and one query-tuple (*Normal, QU-who-what, partners, involved, AKT project, ?*).

4.3.2. Solving question-structure ambiguities

The process of adding the rules above illustrates the ability of quickly handling new question-structure patterns of our knowledge acquisition approach against the ad-hoc approaches [26,35]. The following examples demonstrate the ability of our approach to solve the question-structure ambiguities.

With the question *"which researchers wrote publications related to semantic portals ?"*

[RDR5_: [RDR1_QP: which researchers] [RDR1_Rel: wrote] [RDR1_NP: publications] [RDR3_Rel: related to] [RDR5_NP: semantic portals]]

This question is fired at node (5) which is the last node in the evaluation path (0)-(1)-(2)-(3)-(5). But **R5** gives a wrong conclusion of the "*Normal*" question-structure and one query-tuple (*Normal, QU-whichClass, publications, related to, semantic portals, ?*). We add the following rule **R40** as an exception rule of **R5** to correct the conclusion returned by **R5**:

Rule: R40
(
{RDR5_}
) :left --→ :left.RDR40_ = {category1 ="Normal", category2 = "Normal"}

Condition:
RDR1_QP.hasAnno == QuestionPhrase.category == QU-whichClass

Conclusion:
"*Clause*" question-structure[6] and two query-tuples
(*RDR40_.category1, RDR1_QP.QuestionPhrase.category, RDR1_QP, RDR1_Rel, ?, ?*) and
(*RDR40_.category2, RDR1_QP.QuestionPhrase.category, RDR1_NP, RDR3_Rel, RDR5_NP, ?*)

The extra annotation constraint of *hasAnno* requires that the text covered by an annotation must contain another specified annotation. For example, the additional condition in **R40** only matches the *RDR1_QP* annotation that has a *QuestionPhrase* annotation covering its substring.[7] Additionally, this *QuestionPhrase* annotation must has "*QU-whichClass*" as the value of its *"category"* feature.

In the knowledge base, node (40) containing **R40** is added as the except-child node of node (5). Given the question, the fired node now is node (40); and the conclusion of **R40** produces a correct intermediate representation consisting of the "*Clause*" question-structure and two query-tuples (*Normal, QU-whichClass, researchers, wrote, ?, ?*) and (*Normal, QU-whichClass, publications, related to, semantic portals, ?*).

With another question *"which projects sponsored by eprsc are related to semantic web ?"*

which/WDT projects/NNS sponsored/VBN by/IN eprsc/NN are/VBP related/VBN to/TO semantic/JJ web/NN

[RDR40_: [RDR1_QP: [QuestionPhrase$_{category =QU-whichClass}$: which projects]] [RDR1_Rel: sponsored by] [RDR1_NP: eprsc] [RDR3_Rel: are related to] [RDR5_NP: semantic web]]

[6] A "*Clause*" structure question has two query-tuples where the answer returned for the second query-tuple indicates the missing $Term_2$ attribute in the first query-tuple. See more details of our question-structure definitions in appendix A.

[7] A whole string is also considered as its substring.

The current knowledge base generates an evaluation path (0)-(1)-(2)-(3)-(5)-(40)-(42)-(43) with the fired node (40). However, **R40** returns a wrong conclusion with the "*Clause*" question-structure and two query-tuples (*Normal, QU-whichClass, projects, sponsored, ?, ?*) and (*Normal, QU-whichClass, eprsc, related to, semantic web, ?*) since $Term_1$ cannot be assigned to the instance "eprsc". The following rule **R45** which is a new exception rule of **R40** is added to correct the conclusion given by **R40**:

Rule: R45
(
{RDR40_}
) :left --→ :left.RDR45_ = {category1 ="Normal", category2 = "Normal"}

Condition:
RDR1_Rel.hasAnno == Token.category == VBN

Conclusion:
"*And*" question-structure and two query-tuples
(*RDR45_.category1, RDR1_QP.QuestionPhrase.category, RDR1_QP, RDR1_Rel, RDR1_NP, ?*) and
(*RDR45_.category2, RDR1_QP.QuestionPhrase.category, RDR1_QP, RDR3_Rel, RDR5_NP, ?*)

R45 enables to return a correct intermediate representation element for the question with the "*And*" question-structure and two query-tuples (*Normal, QU-whichClass, projects, sponsored, eprsc, ?*) and (*Normal, QU-whichClass, projects, related to, semantic web, ?*). In the knowledge base, the associated node (45) is attached as the false-child node of node (43).

4.3.3. Porting to other domains

As illustrated in sections 4.3.1 and 4.3.2, using the set of 170 questions from AquaLog [26], we constructed a knowledge base of 59 rules for question analysis. Similarly, in this section, we illustrate the process of adding more exception rules into the knowledge base to handle DBpedia and biomedical test questions.

With the DBpedia test question "*Which presidents of the United States had more than three children ?*"

Which/WDT presidents/NNS of/IN the/DT United/NNP States/NNPS had/VBD more/JJR than/IN three/CD children/NNS

[RDR27_: [RDR10_QP: Which presidents] [Preps: of] [RDR10_NP: the United States] [RDR27_Rel: had more than] [RDR27_NP: three children]]

This question is fired at node (27), however, the conclusion of rule **R27** at node (27) produced an incorrect intermediate representation element for the question. So a new exception rule of **R27** is added to the knowledge base to correct the conclusion returned by **R27** as following:

Rule: R67
(
{RDR10_}
{Verb}
({Token.category==JJR} {Token.string==than} {Token.category==CD}):cp
({Noun}):np
) :left --→ :left.RDR67_ = {category1="Compare", category2 = "UnknRel"}
, :cp.RDR67_Compare = { }
, :np.RDR67_NP = { }

Conclusion:
"*Clause*" question-structure and two query-tuples
(*RDR67_.category1, RDR10_QP.QuestionPhrase.category, ? , RDR67_NP, ?, RDR67_Compare*) and
(*RDR67_.category2, RDR10_QP.QuestionPhrase.category, RDR10_QP, ?, RDR10_NP, ?*)

Given the question, **R67** produces a correct intermediate representation element of the "*Clause*" question-structure and query-tuples *(Compare, QU-whichClass, ?, children, ?, more than three)* and *(UnknRel, QU-whichClass, presidents, ?, United States, ?).*

With the biomedical test question "*List drugs that lead to strokes and arthrosis*"

List/NN drugs/NNS that/WDT lead/VBP to/TO strokes/NNS and/CC arthrosis/NNS

[QuestionPhrase: List drugs] [RDR1_: [RDR1_QP: that] [RDR1_Rel: lead to] [RDR1_NP: strokes and arthrosis]]

This question is fired at node (1), however, **R1** returned an incorrect intermediate representation element. So a new exception rule of **R1** is added to the knowledge base as following:

Rule: R80
(
({QuestionPhrase}):qp
{RDR1_QP} {RDR1_Rel}
({Noun}):np1
{Token.category == CC}
({Noun}):np2
) :left --→ :left.RDR80_ = {category1="Normal", category2="Normal"}
, :qp.RDR80_QP = { }
, :np1.RDR80_NP1 = { }
, :np2.RDR80_NP2 = { }

Condition:
RDR80_QP.hasAnno == Noun

Conclusion:
"*And*" question-structure and two query-tuples (*RDR80_.category1, RDR80_QP.QuestionPhrase.category, RDR80_QP, RDR1_Rel, RDR80_NP1, ?*) and (*RDR80_.category2, RDR80_QP.QuestionPhrase.category, RDR80_QP, RDR1_Rel, RDR80_NP2, ?*)

Given the question, **R80** returns a correct intermediate representation element of the "*And*" question-structure and two query-tuples (*Normal, QU-listClass, drugs, lead to, strokes, ?*) and (*Normal, QU-listClass, drugs, lead to, arthrosis, ?*).

5. Experiments

We separately evaluate the question analysis and answer retrieval components of KbQAS in sections 5.1 and 5.2, respectively. It is because the question analysis component employs our knowledge acquisition approach which is language independent, while the answer retrieval component produces answers from a domain-specific Vietnamese ontology.

5.1. Experiments on analyzing questions

This section is to indicate the abilities of our question analysis approach for quickly building a new knowledge base and easily adapting to a new domain and a new language. We evaluate both our approaches of ad-hoc manner (see section 3.3.3) and knowledge acquisition (see section 4) on Vietnamese question analysis, and then present the experiment of building a knowledge base for processing English questions.

5.1.1. Question Analysis for Vietnamese

We used a training set of 400 various-structure questions generated by four volunteer students to build a Vietnamese knowledge base for question analysis. We then evaluated the quality of the knowledge base on an unseen list of 88 questions related to the VNU University of Engineering and Technology, Vietnam. In this experiment, we also compare both our ad-hoc and knowledge acquisition approaches for question analysis, using the same training set of 400 questions and test set of 88 questions.

Our first approach took about 75 hours to create rules in an ad-hoc manner, as shown in Table 2. In contrast, our second approach took 13 hours to build the knowledge base. However, most of the time was spent for looking at questions to determine the question structures and the phrases which would be extracted to create intermediate representation elements. So the actual time to create rules in the knowledge base was about 5 hours in total.

Table 2
Time to create rules and number of successfully analyzed questions

Type	Time	#questions
Ad-hoc	75 hours	70/88 (79.5%)
Knowledge acquisition	13 hours	74/88 (**84.1%**)

Table 3
Number of exception rules in each layer in our Vietnamese knowledge base for question analysis

Layer	Number of rules
1	26
2	41
3	20
4	4

The knowledge base consists of the default rule and 91 exception rules. Table 3 details the number of exception rules in each layer where every rule in layer n is an exception rule of a rule in layer $n-1$. The only rule which is not an exception rule of any rule is the default rule at layer 0. This indicates that the exception structure is indeed present and even extends to 4 levels.

Table 2 also shows the number of successfully analyzed questions for each approach. By using the knowledge base to resolve ambiguous cases, our knowledge acquisition approach performs better than our ad-hoc approach. Furthermore, Table 4 provides the error sources for our knowledge acquisition approach, in which most errors come from unexpected question structure patterns. This can be rectified by adding more exception rules to the current knowledge base, especially when having a large training set that contains a variety of question structure patterns.

Table 4
Number of incorrectly analyzed questions accounted for the knowledge acquisition approach

Reason	#questions
Unknown structure patterns	12/88
Word segmentation and part-of-speech tagging modules were not trained on question domain	2/88

For another example, our knowledge acquisition approach did not return a correct intermediate representation element for the question *"Vũ Tiến Thành có quê và có mã sinh viên là gì?"* ("what is the hometown and student code of Vu Tien Thanh?") because the existing linguistic processing modules for Vietnamese [42], including word segmentation and part-of-speech tagging, were not trained on the question domain. So these two modules assign the word "quê$_{hometown}$" as an adjective instead of a noun. Thus, "quê$_{hometown}$" is not covered by a *NounPhrase* annotation, leading to an unrecognized structure pattern.

Table 5

Number of rules in the question analysis knowledge bases for Vietnamese (#RV) and English (#RE); number of Vietnamese test questions (#TQ) and number of Vietnamese correctly answered questions (#CA) corresponding to each question-structure type (QST)

QST	#RV	#CA	#TQ	#RE
Definition	2	1	2/2	4
UnknRel	4	4	4/7	6
UnknTerm	7	6	7/7	4
Normal	7	7	7/7	11
Affirm	10	5	5/5	5
Compare	5	0	2/4	8
ThreeTerm	9	7	7/10	6
Affirm_3Term	5	4	4/4	2
And	9	7	8/8	21
Or	23	18	21/24	1
Affirm_MoreTuples	3	1	2/3	1
Clause	6	0	4/5	20
Combine	1	1	1/2	0
Total	91	61	74/88	89

Regarding to an question-structure based evaluation, Table 5 presents the number of rules in the Vietnamese knowledge base and number of test questions, corresponding to each question-structure type. For example, the cell at the second row and the fourth column of Table 5 means that, in 7 test questions tending to have the "*UnknRel*" question-structure, there are 4 test questions correctly analyzed.

5.1.2. *Question Analysis for English*

For the experiment in English, we firstly used a set of 170 English questions,[8] which AquaLog [26] analyzed successfully. These questions are about the Knowledge Media Institute and its research area on Semantic web. Using this question set, we constructed a knowledge base of 59 rules for question analysis. It took 7 hours to build the knowledge base, including 3 hours of actual time to create all rules. We then evaluated the knowledge base using a set of 50 DBpedia test questions[9] from the QALD-1 workshop and another set of 25 biomedical test questions[10] from the QALD-4 workshop.

Table 6

Testing results of the knowledge base of 59 rules for question analysis on DBpedia and biomedical domains

Factor	DBpedia	Biomedical
Successfully processed	24/50	9/25
Unknown structure patterns	18/50	9/25
Incorrect word segmentation	3/50	3/25
Incorrect Part-of-speech tagging	5/50	4/25

Table 6 presents evaluation results on analyzing the test questions from the DBpedia and biomedical domains, using the knowledge base of 59 rules for question analysis. It is not surprising that most errors come from unknown question structure patterns. Furthermore, just as in Vietnamese, the existing linguistic processing modules in the GATE framework [11], including the English tokenizer and Part-of-speech tagger, are also error sources, leading to unrecognized structure patterns. For example, such questions as "*Which U.S. states possess gold minerals ?*" and "*Which drugs have a water solubility of 2.78e-01mg/mL ?*" are tokenized into "*Which U . S . states possess gold minerals ?*" and "*Which drugs have a water solubility of 2 . 78 e- 01 mg / mL ?*", respectively. In addition, such other questions as "*Which river does the Brooklyn Bridge cross ?*", "*Which states border Utah?*" or "*Which experimental drugs interact with food ?*" are tagged with noun labels for the words "cross", "border" and "interact" instead of verb labels.

Table 7

Testing results of the English knowledge base of 90 rules for question analysis on DBpedia and biomedical domains

Factor	DBpedia	Biomedical
Successfully processed	47/50	21/25
Unknown structure patterns	0/50	0/25
Incorrect word segmentation	3/50	3/25
Incorrect Part-of-speech tagging	0/50	1/25

To correct the question analysis errors on the two sets of test questions, we spent 5 further hours to add 31 exception rules to the knowledge base. Finally, in total 12 hours, we constructed a knowledge base of

[8] http://technologies.kmi.open.ac.uk/aqualog/examples.html

[9] http://greententacle.techfak.uni-bielefeld.de/~cunger/qald/1/data/dbpedia-test-questions.xml

[10] http://greententacle.techfak.uni-bielefeld.de/~cunger/qald/4/data/qald-4_biomedical_test_questions.xml

90 rules for English question analysis, including the default rule and 89 exception rules. The new evaluation results of question analysis on the DBpedia and biomedical domains are presented in Table 7.

Table 8 shows the number of exception rules in each exception layer of the knowledge base while the number of rules corresponding to each question-structure type is presented Table 5.

Table 8

Number of exception rules in layers in our English knowledge base

Layer	Number of rules
1	10
2	21
3	31
4	20
5	7

As the intermediate representation in KbQAS is different to AquaLog, it is difficult to directly compare our knowledge acquisition approach with the ad-hoc question analysis approach in AquaLog on the English domain. However, this experiment on English questions shows the abilities to quickly build a new knowledge base and easily adapt to a new domain and a new language of our knowledge acquisition approach for question analysis.

As illustrated in section 4.3, this experiment also presented a process of building a knowledge base for question analysis without any concept or entity type information. However, we found that the concept or entity type information inside noun phrases is useful and somehow can help to reduce the ambiguities in question structure patterns. When adapting our knowledge acquisition approach for question analysis to anther target domain (or language), we can simply use the heuristics presented in section 3.3.2 and a dictionary to determine whether a noun phrase is a concept or entity type. The dictionary can be (automatically) constructed by extracting concepts from the target domain and theirs synonyms from available semantic lexicons such as WordNet [17].

5.2. Experiment on answering Vietnamese questions

To evaluate KbQAS by specifying in the Answer retrieval component, we used the ontology modeling the organizational structure of the VNU University of Engineering and Technology, as mentioned in the section 3.2, as target domain. This ontology was manually constructed by using the Protégé platform [19]. From the list of 88 questions, as mentioned in section 5.1.1, we employed 74 questions which were successfully analyzed by the question analysis component.

Table 9

Questions successfully answered

Type	# questions
No interaction with users	30/74
With interactions with users	31/74
Overall	61/74 (**82.4%**)

The performance result is presented in Table 9. The answer retrieval component produces correct answers to 61 questions over 74 questions, obtaining a promising accuracy of 82.4%. The number of correctly answered questions corresponding with each question-structure type can be found in the third column of Table 5. Out of those, 30 questions can be answered automatically without interaction with users. In addition, 31 questions are correctly answered with the help from the users to handle ambiguity cases, as illustrated in the first example in section 3.4.

Table 10

Questions with unsuccessful answers

Type	# questions
Ontology mapping errors	6/74
Answer extraction errors	7/74

Table 10 gives the limitations that will be handled in future KbQAS versions. The errors raised by the Ontology mapping module are because the target ontology construction lacked a full domain-specific conceptual coverage and some relationships between concept pairs. So specific terms or relations in query-tuples cannot be mapped or incorrectly mapped to the corresponding elements in the target ontology to produce the Ontology-tuples. Furthermore, the Answer extraction module fails to extract the answers to 7 questions because: (i) Dealing with questions having the *"Compare"* question-structure involves specific services. For example, handling the question *"sinh viên nào có điểm trung bình cao nhất khoa công nghệ thông tin?"* (which student has the highest grade point average in faculty of Information Technology?) requires a comparison mechanism to rank students according to their GPA. (ii) In terms of four "*Clause*" structure questions and one "*Affirm_MoreTuples*" structure question that KbQAS failed to return answers (see Table 5), combining their sub-questions triggers complex inference tasks and bugs which are difficultly to handle in the current KbQAS version.

6. Conclusion and future work

In this paper, we described the first ontology-based question answering system for Vietnamese namely KbQAS. KbQAS contains two components of Natural language question analysis and Answer retrieval. The two components are connected by an intermediate representation element capturing the semantic structure of any input question, facilitating the matching process to a target ontology to produce answer. Experimental results of KbQAS on a wide range of questions are promising. Specifically, the answer retrieval module achieves an accuracy of 82.4%.

In addition, we proposed a question analysis approach for systematically building a knowledge base of rules to convert the input question into an intermediate representation element. Our approach allows systematic control of interactions between rules and keeping consistency among them. We believe that our approach is important especially for under-resourced languages where annotated data is not available. Our approach could be combined nicely with the process of annotating corpus where, on top of assigning a label or a representation to a question, the experts just have to add one more rule to justify their decision. Incrementally, an annotated corpus and a rule-based system can be obtained simultaneously. Furthermore, our approach can be applied to open domain question answering where the technique requires an analysis to turn the input question to an explicit representation of some sort. Obtaining a question analysis accuracy of 84.1% on Vietnamese questions and taking 12 hours to build a knowledge base of 90 rules for analyzing English questions, the question analysis experiments show that our approach enables individuals to easily build a new knowledge base or adapt an existing knowledge base to a new domain or a new language.

In the future, we will extend KbQAS to be an open domain question answering system which can answer various questions over Linked Open Data such as DBpedia or YAGO. In addition, it would be interesting to investigate the process of building a knowledge base for question analysis, which directly converts the input questions into queries (e.g. SPARQL queries) on the Linked Open Data.

Acknowledgment

This work is partially supported by the Research Grant No. QG.14.04 from the Vietnam National University, Hanoi.

Appendix

A. Definitions of question-structure types

We define question-structures types: *Normal, UnknTerm, UnknRel, Definition, Affirm, ThreeTerm, Affirm_3Term, Affirm_MoreTuples, Compare, And, Or, Combine, Clause* as following:

- A “*Normal*” structure question has only one query-tuple in which $Term_3$ is missing.
- An “*UnknTerm*” structure question has only one query-tuple in which $Term_1$ and $Term_3$ are missing.
- An “*UnknRel*” structure question has only one query-tuple in which $Relation$ and $Term_3$ are missing. For example, the question “List all the publications in knowledge media institute” has one query-tuple *(UnknRel, QU-listClass, publications, ?, knowledge media institute, ?)*.
- A “*Definition*” structure question has only one query-tuple which lacks $Term_1$, $Relation$ and $Term_3$. For example, the question “what are research areas?” has one query-tuple *(Definition, QU-who-what, ?, ?, research areas, ?)*.
- An “*Affirm*” structure question is the question which belongs to one of three types “*Normal*”, “*UknnRel*” and “*UnknTerm*”, and has the “*YesNo*” question-category. For example, the question “Is Tran Binh Giang a Phd student?” has the “*Affirm*” question-structure and one query-tuple *(UnknRel, YesNo, Phd student, ?, Tran Binh Giang, ?)*
- A “*ThreeTerm*” structure question has only one query-tuple where $Term_1$ or $Relation$ could be missing. An example for this structure type is illustrated in Figure 8.
- An “*Affirm_3Term*” structure question is the question which belongs to the “*ThreeTerm*” and has the “*YesNo*” question-category. For example, the question *“số lượng sinh viên học lớp K50 khoa học máy tính là 45 phải không?”* (“45 is the number of students studying in K50 computer science course, is not it?”) has the “*Affirm_3Term*” question structure and one query-tuple (*ThreeTerm, ManyClass, sinh viên*$_{student}$, *học*$_{study}$, *lớp K50 khoa học máy tính*$_{K50computersciencecourse}$, *45*).
- An “*Affirm_MoreTuples*” structure question has more than one query-tuple and belongs to the “*YesNo*” question-category. For example, the question *“tồn tại sinh viên có quê ở Hà Tây và học khoa toán phải không ?”* (“is there some student having hometown in Hatay and studying in faculty of Mathematics?”) has the “*Affirm_MoreTuples*” question

structure and two query-tuples *(Normal, YesNo, sinh viên*$_{student}$*, có quê*$_{have\ hometown}$*, Hà Tây*$_{Hatay}$*, ?)* and *(Normal, YesNo, sinh viên*$_{student}$*, học*$_{study}$*, khoa Toán*$_{faculty\ of\ Mathematics}$*, ?)*.

- A "*Compare*" structure question is the question which belongs to one of three types "*Normal*", "*UnknRel*" and "*UnknTerm*", and it contains a comparison phrase which is detected by the preprocessing module. In this case, $Term_3$ is used to hold the comparison information. For example, the question *"sinh viên nào có điểm trung bình cao nhất khoa công nghệ thông tin?"* ("which student has the highest grade point average in faculty of Information Technology?") has the "*Compare*" query-structure and one query-tuple (*Normal, Entity, sinh viên*$_{student}$*, điểm trung bình*$_{grade\ point\ average}$*, khoa công nghệ thông tin*$_{faculty\ of\ Information\ Technology}$*, cao nhất*$_{highest}$).
- An "*And*" or "*Or*" structure question contains the word *"mà*$_{and}$*"* (*"và*$_{and}$*"*) or *"hoặc*$_{or}$*"*, respectively, and it has more than one query-tuple (i.e. two or more sub-questions). The "*And*" type returns the final answer as an intersection (i.e. overlap) of the answers for the sub-questions, while the "*Or*" type returns the final answer as an union of the answers for the sub-questions.

For example, the question "which projects are about ontologies and the semantic web?" has the "*And*" question-structure and two query-tuples *(UnknRel, QU-whichClass, projects, ?, ontologies, ?)* and *(UnknRel, QU-whichClass, projects, ?, semantic web, ?)*.

The question "which publications are in knowledge media institute related to compendium?" has the "*And*" question-structure and two query-tuples *(UnknRel, QU-whichClass, publications, ?, knowledge media institute, ?)* and *(Normal, QU-whichClass, publications, related to, compendium, ?)*.

The question "who is interested in ontologies or in the semantic web?" has the "*Or*" question-structure and two query-tuples *(UnknTerm, QU-who-what, ?, interested, ontologies, ?)* and *(UnknTerm, QU-who-what, ?, interested, semantic web, ?)*.

However, some questions such as the question *"Phạm Đức Đăng học trường đại học nào và được hướng dẫn bởi ai?"* ("Which university does Pham Duc Dang study in and who tutors him?") contains *"và*$_{and}$*"*, but it will has the "*Or*" question-structure and two query-tuples (*Normal, Entity, trường đại học*$_{university}$*, học*$_{study}$*, Phạm Đức Đăng*$_{Pham\ Duc\ Dang}$ *, ?*) and (*UnknTerm, Who, ?, hướng dẫn*$_{tutor}$*, Phạm Đức Đăng*$_{Pham\ Duc\ Dang}$*, ?*).

- A "*Combine*" structure question is constructed from two or more independent sub-questions. Unlike the "Or" structure type, the query-tuples in the " *Combine*" type do not share the same term or $Relation$. For example, the question *"Ai có quê quán ở Hà Tây và ai học khoa công nghệ thông tin?"* ("who has hometown in Hatay, and who study in faculty of Information Technology?") has the "*Combine*" question-structure and two query-tuples (*UnknTerm, Who, ?, có quê quán*$_{has\ hometown}$*, Hà Tây*$_{Hatay}$*, ?*) and (*UnknTerm, Who, ?, học*$_{study}$*, khoa công nghệ thông tin*$_{faculty\ ofInformation\ Technology}$*, ?*).
- A "*Clause*" structure question has two query-tuples, where the answer returned for the second query-tuple indicates the missing $Term_2$ attribute in the first query-tuple. For example, the question *"số lượng sinh viên học lớp K50 khoa học máy tính lớn hơn 45 phải không ?"*[11] (the number of students studying in K50 computer science course is higher than 45, is not it?) has the "*Clause*" question-structure and two query-tuples *(Compare, YesNo, 45, ?, ?, lớn hơn*$_{higher\ than}$*)* and *(Normal, ManyClass, sinh viên*$_{student}$*, học*$_{study}$*, lớp K50 khoa học máy tính*$_{K50\ computer\ science\ course}$*, ?)*. Another example of this "*Clause*" structure type is presented in section 4.3.2.

In general, $Term_1$ represents a concept, excluding cases of *Affirm*, *Affirm_3Term* and *Affirm_MoreTuples*. In addition, $Term_2$ and $Term_3$ represent entities (i.e. objects or instances), excluding the cases of "*Definition*" and "*Compare*".

B. Definitions of Vietnamese question-categories

In KbQAS, question is classified into one of the following classes of *HowWhy, YesNo, What, When, Where, Who, Many, ManyClass, List*, and *Entity*. To identify question categories, we specify a number of JAPE grammars using the *NounPhrase* annotations and the question-word information given by the preprocessing module.

- A *HowWhy*-category question refers a cause or a method, containing a *TokenVn* annotation covering such string as *"tại sao*$_{why}$*"* or *"vì sao*$_{why}$*"* or *"thế nào*$_{how}$*"* or *"là như thế nào*$_{how}$*"*. This is similar to *Why*-question or *How is/are* question in English.
- A *YesNo*-category question requires a true or false answer, containing a *TokenVn* annotation cov-

[11] This is the case of our system failing to correctly analyze due to an unknown structure pattern.

ering such string as *“có đúng là*$_{is\ that}$*”* or *“đúng không*$_{are\ those}$*”* or *“phải không*$_{are\ there}$*”* or *“có phải là*$_{is\ this}$*”* .

- A *What*-category question contains a *TokenVn* annotation covering such string as *“cái gì*$_{what}$*”* or *“là gì*$_{what}$*”* or *“là những cái gì*$_{what}$*”*. This question type is similar to *What is/are* question type in English.
- A *When*-category question contains a *TokenVn* annotation covering such string as *“khi nào*$_{when}$*”* or *“vào thời gian nào*$_{which\ time}$*”* or *“lúc nào*$_{when}$*”* or *“ngày nào*$_{which\ date}$*”*.
- A *Where*-category question contains a *TokenVn* annotation covering such string as *“ở nơi nào*$_{where}$*”* or *“là ở nơi đâu*$_{where}$*”* or *“ở chỗ nào*$_{where}$*”*.
- A *Who*-category question contains a *TokenVn* annotation covering such string as *“là những ai*$_{who}$*”* or *“là người nào*$_{who}$*”* or *“những ai*$_{who}$*”*.
- A *Many*-category question contains a *TokenVn* annotation covering such string as *“số lượng*$_{how\ many}$*”* or *“là bao nhiêu*$_{how\ much|many}$*”* or *“bao nhiêu*$_{how\ much|many}$*”*. This question type is similar to *How much/many is/are* question type in English.
- A *ManyClass*-category question contains a *TokenVn* annotation covering such string as *“số lượng*$_{how\ many}$*”* or *“là bao nhiêu*$_{how\ much|many}$*”* or *“bao nhiêu*$_{how\ much|many}$*”*, followed by a *NounPhrase* annotation. This type is similar to *How many NounPhrase*-question type in English.
- An *Entity*-category question contains a *NounPhrase* annotation followed by a *TokenVn* annotation covering such string as *“nào*$_{which}$*”* or *“gì*$_{what}$*”*. This type is similar to *which/what NounPhrase*-question type in English.
- A *List*-category question contains a *TokenVn* annotation covering such string as *“cho biết*$_{give}$*”* or *“chỉ ra*$_{show}$*”* or *“kể ra*$_{tell}$*”*, or *“tìm*$_{find}$*”* or *“liệt kê*$_{list}$*”*, followed by a *NounPhrase* annotation.

References

[1] I. Androutsopoulos, G. Ritchie, and P. Thanisch. Masque/sql– An Efficient and Portable Natural Language Query Interface for Relational Databases. In *Proceedings of the 6th International Conference on Industrial & Engineering Applications of Artificial Intelligence and Expert Systems*, pages 327–330, 1993.

[2] I. Androutsopoulos, G. Ritchie, and P. Thanisch. Natural language interfaces to databases - an introduction. *Natural Language Engineering*, 1(1):29–81, 1995.

[3] P. Atzeni, R. Basili, D. H. Hansen, P. Missier, P. Paggio, M. T. Pazienza, and F. M. Zanzotto. Ontology-Based Question Answering in a Federation of University Sites: The MOSES Case Study. In *Proceedings of 9th International Conference on Applications of Natural Languages to Information Systems*, pages 413–420, 2004.

[4] R. Basili, D. H. Hansen, P. Paggio, M. T. Pazienza, and F. M. Zanzotto. Ontological resources and question answering. In *HLT-NAACL 2004: Workshop on Pragmatics of Question Answering*, pages 78–84, 2004.

[5] J. Berant and P. Liang. Semantic Parsing via Paraphrasing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1415–1425, 2014.

[6] J. Berant, A. Chou, R. Frostig, and P. Liang. Semantic Parsing on Freebase from Question-Answer Pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1533–1544, 2013.

[7] D. Bernhard and I. Gurevych. Answering Learners' Questions by Retrieving Question Paraphrases from Social Q&A Sites. In *Proceedings of the Third ACL Workshop on Innovative Use of NLP for Building Educational Applications*, pages 44–52, 2008.

[8] R. D. Burke, K. J. Hammond, V. A. Kulyukin, S. L. Lytinen, N. Tomuro, and S. Schoenberg. Question Answering from Frequently Asked Question Files: Experiences with the FAQ FINDER System. *AI Magazine*, 18(2):57–66, 1997.

[9] P. Cimiano, P. Haase, J. Heizmann, M. Mantel, and R. Studer. Towards Portable Natural Language Interfaces to Knowledge Bases - The Case of the ORAKEL System. *Data & Knowledge Engineering*, 65(2):325–354, 2008.

[10] P. Compton and R. Jansen. A philosophical basis for knowledge acquisition. *Knowledge Aquisition*, 2(3): 241–257, 1990.

[11] H. Cunningham, D. Maynard, K. Bontcheva, and V. Tablan. GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications. In *Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics*, pages 168–175, 2002.

[12] D. Damljanovic, V. Tablan, and K. Bontcheva. A Text-based Query Interface to OWL Ontologies. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation*, pages 205–212, 2008.

[13] D. Damljanovic, M. Agatonovic, and H. Cunningham. Natural Language Interfaces to Ontologies: Combining Syntactic Analysis and Ontology-based Lookup Through the User Interaction. In *Proceedings of the 7th International Conference on The Semantic Web: Research and Applications - Volume Part I*, pages 106–120, 2010.

[14] Q. B. Diep. *Ngữ pháp tiếng Việt (Grammar of Vietnamese language)*. Vietnam Education Publishing

House, 2005.

[15] T. Dong, U. Furbach, I. Glöckner, and B. Pelzer. A Natural Language Question Answering System as a Participant in Human Q&A Portals. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence*, pages 2430–2435, 2011.

[16] A. Fader, L. Zettlemoyer, and O. Etzioni. Paraphrase-Driven Learning for Open Question Answering. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1608–1618, 2013.

[17] C. D. Fellbaum. *WordNet: An Electronic Lexical Database*. MIT Press, 1998.

[18] U. Furbach, I. Glöckner, and B. Pelzer. An Application of Automated Reasoning in Natural Language Question Answering. *AI Communications*, 23:241–265, 2010.

[19] J. H. Gennari, M. A. Musen, R. W. Fergerson, W. E. Grosso, M. Crubzy, H. Eriksson, N. F. Noy, and S. W. Tu. The Evolution of Protégé: An Environment for Knowledge-Based Systems Development. *International Journal of Human-Computer Studies*, 58:89–123, 2002.

[20] S. Harabagiu, D. Moldovan, M. Paşca, R. Mihalcea, M. Surdeanu, Z. Bunescu, R. Girju, V. Rus, and P. Morarescu. Falcon: Boosting knowledge for answer engines. In *Proceedings of the Ninth Text REtrieval Conference*, pages 479–488, 2000.

[21] L. Hirschman and R. Gaizauskas. Natural Language Question Answering: The View from Here. *Natural Language Engineering*, 7(4):275–300, 2001.

[22] V. Jijkoun and M. de Rijke. Retrieving Answers from Frequently Asked Questions Pages on the Web. In *Proceedings of the 14th ACM International Conference on Information and Knowledge Management*, pages 76–83, 2005.

[23] B. Katz. Annotating the World Wide Web using Natural Language. In *Proceedings of the 5th RIAO Conference on Computer Assisted Information Searching on the Internet*, pages 136–159, 1997.

[24] O. Kolomiyets and M.-F. Moens. A Survey on Question Answering Technology from an Information Retrieval Perspective. *Information Sciences*, 181(24): 5412–5434, 2011.

[25] Z. Liu, X. Qiu, L. Cao, and X. Huang. Discovering Logical Knowledge for Deep Question Answering. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, pages 1920–1924, 2012.

[26] V. Lopez, V. Uren, E. Motta, and 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 (2):72–105, 2007.

[27] V. Lopez, V. Uren, M. Sabou, and E. Motta. Is Question Answering Fit for the Semantic Web?: A Survey. *Semantic Web*, 2(2):125–155, 2011.

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

[29] P. Martin, D. E. Appelt, B. J. Grosz, and F. Pereira. TEAM: An Experimental Transportable Natural-language Interface. In *Proceedings of 1986 ACM Fall Joint Computer Conference*, pages 260–267, 1986.

[30] D. L. McGuinness. Question Answering on the Semantic Web. *IEEE Intelligent Systems*, 19(1):82–85, 2004.

[31] A. C. Mendes and L. Coheur. When the answer comes into question in question-answering: survey and open issues. *Natural Language Engineering*, 19(1):1–32, 2013.

[32] M. Minock. C-Phrase: A System for Building Robust Natural Language Interfaces to Databases. *Data & Knowledge Engineering*, 69(3):290–302, 2010.

[33] D. Moldovan, S. Harabagiu, R. Girju, P. Morarescu, F. Lacatusu, A. Novischi, A. Badulescu, and O. Bolohan. LCC Tools for Question Answering. In *Proceedings of the 11th Text REtrieval Conference*, 2002.

[34] A. K. Nguyen and H. T. Le. Natural Language Interface Construction Using Semantic Grammars. In *Proceedings of the 10th Pacific Rim International Conference on Artificial Intelligence: Trends in Artificial Intelligence*, pages 728–739, 2008.

[35] D. Q. Nguyen, D. Q. Nguyen, and S. B. Pham. A Vietnamese Question Answering System. In *Proceedings of the 2009 International Conference on Knowledge and Systems Engineering*, pages 26–32, 2009.

[36] D. Q. Nguyen, D. Q. Nguyen, and S. B. Pham. Systematic Knowledge Acquisition for Question Analysis. In *Proceedings of the International Conference Recent Advances in Natural Language Processing 2011*, pages 406–412, 2011.

[37] D. Q. Nguyen, D. Q. Nguyen, S. B. Pham, and D. D. Pham. Ripple Down Rules for Part-of-Speech Tagging. In *Proceedings of the 12th international conference on Computational linguistics and intelligent text processing - Volume Part I*, pages 190–201, 2011.

[38] D. Q. Nguyen, D. Q. Nguyen, and S. B. Pham. KbQAS: A Knowledge-based QA System. In *Proceedings of the ISWC 2013 Posters & Demonstrations Track*, pages 109–112, 2013.

[39] D. T. Nguyen and T. P.-M. Nguyen. A Question Answering Model Based Evaluation for OVL (Ontology for Vietnamese Language). *International Journal of Computer Theory and Engineering*, 3(3), 2011.

[40] D. T. Nguyen, T. D. Hoang, and S. B. Pham. A Vietnamese Natural Language Interface to Database. In *Proceedings of the 2012 IEEE Sixth International Conference on Semantic Computing*, pages 130–133, 2012.

[41] A. Peñas, B. Magnini, P. Forner, R. Sutcliffe, A. Rodrigo, and D. Giampiccolo. Question answering at the cross-language evaluation forum 2003-2010. *Language*

Resources and Evaluation, 46(2):177–217, 2012.

[42] D. D. Pham, G. B. Tran, and S. B. Pham. A Hybrid Approach to Vietnamese Word Segmentation Using Part of Speech Tags. In *Proceedings of the 2009 International Conference on Knowledge and Systems Engineering*, pages 154–161, 2009.

[43] S. B. Pham and A. Hoffmann. Efficient Knowledge Acquisition for Extracting Temporal Relations. In *Proceedings of the 17th European Conference on Artificial Intelligencesssss*, pages 521–525, 2006.

[44] T. Phan and T. Nguyen. Question semantic analysis in Vietnamese QA System. In *Edited book "Advances in Intelligent Information and Database Systems" of The 2nd Asian Conference on Intelligent Information and Database Systems*, pages 29–40, 2010.

[45] A.-M. Popescu, O. Etzioni, and H. Kautz. Towards a Theory of Natural Language Interfaces to Databases. In *Proceedings of the 8th International Conference on Intelligent User Interfaces*, pages 149–157, 2003.

[46] D. Richards. Two decades of ripple down rules research. *Knowledge Engineering Review*, 24(2):159–184, 2009.

[47] F. Rinaldi, J. Dowdall, K. Kaljurand, M. Hess, and D. Mollá. Exploiting paraphrases in a Question Answering system. In *Proceedings of the second international workshop on Paraphrasing - Volume 16*, pages 25–32, 2003.

[48] S. Silakari, M. Motwani, and N. Nihalani. Natural language Interface for Database: A Brief review. *IJCSI International Journal of Computer Science Issues*, 8: 600–608, 2011.

[49] N. Stratica, L. Kosseim, and B. C. Desai. NLIDB Templates for Semantic Parsing. In *Proceedings of the 8th International Conference on Applications of Natural Language to Information Systems*, pages 235–241, 2003.

[50] M. Templeton and J. Burger. Problems in natural-language interface to DBMS with examples from EUFID. In *Proceedings of the first conference on Applied natural language processing*, pages 3–16, 1983.

[51] C. A. Thompson, R. J. Mooney, and L. R. Tang. Learning to Parse Natural Language Database Queries into Logical Form. In *Proceedings of the ML-97 Workshop on Automata Induction, Grammatical Inference, and Language Acquisition*, 1997.

[52] M.-V. Tran, D.-T. Le, X.-T. Tran, and T.-T. Nguyen. A Model of Vietnamese Person Named Entity Question Answering System. In *Proceedings of the 26th Pacific Asia Conference on Language, Information, and Computation*, pages 325–332, 2012.

[53] C. Unger and P. Cimiano. Pythia: Compositional Meaning Construction for Ontology-Based Question Answering on the Semantic Web. In *Proceedings of the 16th International Conference on Applications of Natural Language to Information Systems*, pages 153–160, 2011.

[54] C. Unger, L. Bühmann, J. Lehmann, A.-C. Ngonga Ngomo, D. Gerber, and P. Cimiano. Template-based Question Answering over RDF Data. In *Proceedings of the 21st International Conference on World Wide Web*, pages 639–648, 2012.

[55] B. Van Durme, Y. Huang, A. Kupść, and E. Nyberg. Towards Light Semantic Processing for Question Answering. In *Proceedings of the HLT-NAACL 2003 workshop on Text meaning - Volume 9*, pages 54–61, 2003.

[56] M. Vargas-Vera and E. Motta. An Ontology-Driven Similarity Algorithm. Technical report, Knowledge Media Institute, The Open University, 2004.

[57] E. M. Voorhees. Overview of the TREC-9 Question Answering Track. In *Proceedings of the 9th Text Retrieval Conference*, pages 71–80, 2000.

[58] E. M. Voorhees. The TREC question answering track. *Natural Language Engineering*, 7(4):361–378, 2001.

[59] E. M. Voorhees. Overview of the TREC 2002 Question Answering Track. In *Proceedings of the 11th Text REtrieval Conference*, pages 115–123, 2002.

[60] D. L. Waltz. An English Language Question Answering System for a Large Relational Database. *Communications of the ACM*, 21(7):526–539, 1978.

[61] C. Wang, M. Xiong, Q. Zhou, and Y. Yu. PANTO: A Portable Natural Language Interface to Ontologies. In *Proceedings of the 4th European conference on The Semantic Web: Research and Applications*, pages 473–487, 2007.

[62] B. Webber and N. Webb. Question Answering. In *The Handbook of Computational Linguistics and Natural Language Processing*, pages 630–654. 2010.

[63] W. A. Woods, R. Kaplan, and N. B. Webber. The LUNAR Sciences Natural Language Information System: Final Report. Technical Report BBN Report No. 2378, Bolt Beranek and Newman, 1972.

[64] D. H. Younger. Recognition and parsing of context-free languages in time n3. *Information and Control*, 10(2): 189 – 208, 1967.

[65] H.-J. Zeng, Q.-C. He, Z. Chen, W.-Y. Ma, and J. Ma. Learning to Cluster Web Search Results. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 210–217, 2004.

[66] S. Zhao, M. Zhou, and T. Liu. Learning Question Paraphrases for QA from Encarta Logs. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, pages 1795–1801, 2007.

[67] Z. Zheng. AnswerBus question answering system. In *Proceedings of the Second International Conference on Human Language Technology Research*, pages 399–404, 2002.