Ripple Down Rules for Question Answering

Dat Quoc Nguyen [a,c,*], Dai Quoc Nguyen [b,c] and Son Bao Pham [c]
[a] *Department of Computing, Faculty of Science and Engineering, 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: {datnq,dainq,sonpb}@vnu.edu.vn

Abstract: Recent years have witnessed a new trend on building ontology-based question answering systems, that is to use semantic web information to provide more precise answers to users' queries. However, these systems are mostly designed for English, therefore, we introduce in this paper such a system for Vietnamese, that is, to the best of our knowledge, the first one made for Vietnamese. Different from most of previous works, we propose an approach that systematically builds a knowledge base of grammar rules for processing each input question into an intermediate representation element. Then we take this element with respect to a target ontology by applying concept-matching techniques for returning an answer. Experimental results show that the performance of the system on a wide range of Vietnamese questions is promising with accuracies of 84.1% and 82.4% for analyzing question and retrieving answer, respectively. Furthermore, our approach to the question analysis 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

The availability of online information accessible to human users often requires more support from advanced information retrieval technologies to obtain expected information. This brings new challenges to the construction of information retrieval systems such as search engines and question answering (QA) systems. Most current search engines take an user's query and returns 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 exact answers to the users' questions without involving the scanning process. It is also desirable to allow users to specify questions using natural language expressions rather than the keyword-based approach.

In general, an open-domain QA system aims to potentially answer any user's question, whereas a restricted-domain QA system only handles questions related to a specific domain. Specifically, while traditional restricted-domain systems make use of relational databases to represent target domains, the recent ones utilize knowledge bases such as ontologies as target domains [28] to take advantages of recent advances in semantic web. Thus, semantic markups can be used to add meta-information to provide precise answers to complex natural language questions. This is an avenue which has not been actively explored for Vietnamese.

In this paper, we introduce a **k**nowledge-**b**ased QA **s**ystem for Vietnamese (KbQAS), the first ontology-based QA system for Vietnamese. Our KbQAS system consists of question analysis and answer retrieval components. The front-end question analysis component uses a knowledge base of grammar rules for processing input questions, the back-end answer retrieval component is responsible for making sense of the input query with respect to a target ontology. The association between these two components is an intermediate representation element which contains some properties of questions: construction type, category, keywords and

[*] Corresponding author.

semantic constraints between them for capturing the semantic structure of the question.

The **key innovation** of our system propose a knowledge acquisition approach for systematically constructing a knowledge base for analyzing natural language questions. To translate a natural language question into an explicit representation in QA systems, most previous works so far have used rule-based approaches to the best of our knowledge. Owing to their representation complexity and the variety of question-structure types, manual creation of rules in an ad-hoc manner is very expensive in terms of time, effort and error-prone. For example, many rule-based methods such as to handle English questions [25], to process Vietnamese questions presented in our first KbQAS version [34] manually define a list of pattern structures to analyze questions. As rules are created in an ad-hoc manner, those approaches share common difficulties in managing the interaction between rules and keeping consistency among them. In our approach, however, we utilize Single Classification Ripple Down Rules (SCRDR) [8,45] knowledge acquisition methodology to acquire rules in a systematic manner where the consistency between rules is maintained and the unintended interaction among rules is avoided. Our method allows an easy adaptation to a new domain and a new language and saves a lot of time and effort of human experts.

The paper is organized as follows: we revise related works in section 2. We describe our overall system architecture and our knowledge acquisition approach for question analysis in section 3 and section 4, respectively. We evaluate our KbQAS system 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 [18,62,29]. For example, such systems as START [20], FAQ Finder [6] and AnswerBus [66] try to answer questions over the Web. START uses natural language annotations to extract answers in the use of subject-relation-object form whilst AnswerBus combines 5 search engines and dictionaries to extract Web pages related to input factoid questions and then returns the most suitable sentences containing answers. Using frequently-asked question files as its knowledge base, meanwhile, FAQ Finder's goal is to identify the similarity between user questions and question/answer pairs from FAQ files. Question-paraphrase recognition is then considered as one of the important tasks in QA with many proposed approaches based on statistics, machine learning as well as knowledge representation and reasoning as mentioned in [5,19,46,65].

Since aroused by the QA track of the Text Retrieval Conference [58] and the multilingual QA track of CLEF [40], there is a dramatic increase in the number of open-domain QA systems from the information retrieval perspective [21]. For instance, Falcon [17] adapting the similar architecture of its ancient Lasso system [32] achieved the highest results in the TREC-9 QA competition [56] at 58% for short answers and 78% for long answers. The innovation of Falcon focused on proposing a methodology for boosting knowledge in exploiting WordNet [14]. In the QA track of TREC 2002 [59], PowerAnswer [31] was cited as the most powerful system with the result at 85.6% obtained by relying on a deep linguistic analysis.

2.2. *Traditional restricted-domain question answering*

Traditional restricted-domain QA systems usually linked to relational databases 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 input question into an expression in the corresponding database query language. For instance, Sneiders [48] presented a NLIDB system where the input is converted into SQL query by using defined templates that contain entity slots – free space for data instances representing the primary concepts of the question. In syntactic-based NLIDB systems, the user's question is syntactically transferred into a parsed tree, and the tree is directly converted into an expression of some database query language. LUNAR [63] is a typical example of this approach. However, it is difficult to create translating rules that will directly transform the tree into the query expression.

Later NLIDBs as Planes [60], Eufid [50], C-Phrase [30], the system proposed by Nguyen and Le [33] use semantic grammar to analyze questions. These systems still respond to users' questions by parsing the input into a syntax tree and mapping the tree to a database query, in which the semantic grammar's categories do not correspond to syntactic concepts [2]. Semantic grammars consist of hard-wired knowledge orienting specific domain, hence, those NLIDB systems need to develop new grammars whenever porting to new knowledge domains. For example, the PRECISE system [44] maps the natural language question to an unique semantic interpretation by analyzing some lexicons and semantic constraints. PRECISE showed a high precision of about 80% for a list of hundreds English questions. However, PRECISE requires all tokens in input questions to be distinct and appear in its lexicon. Stratica et al. [49] described a template-based system to translate the English question into SQL query by matching the syntactic parsed tree of the question with a set of fixed semantic templates.

Additionally, systems like TEAM [27] and Masque/sql [1] use semantic information to analyze questions by utilizing syntactic-semantic interpretation rules driving logical forms. These systems firstly transform the input 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. The use of logic languages is to possibly adapt to other domains as well as different query languages [47]. Meanwhile, there are many open-domain systems also using logical forms to process input questions such as in [31,54,15,13,23].

2.3. *Ontology-based question answering*

Considered as a knowledge representation of a set of concepts and their relations due to a specific domain, an ontology could provide semantic information to solve ambiguities, interpret and answer user questions in terms of QA [24]. Discussion on an approach to possibly build an ontology-based QA system can be found in [4]. The approach was then applied to construct the MOSES system [3] in focusing on question analysis. Following is some typical ontology-based QA systems.

Aqualog [25] performs semantic and syntactic analysis of the input question through the use of processing resources provided by the GATE framework [9] including word segmentation, sentence segment, part-of-speech tagging. When a question is asked, the task of its Linguistic Component is to transfer the natural language question to a Query-Triple with the following format (*generic term, relation, second term*). Through the use of JAPE grammars in GATE, AquaLog identifies terms and their relationship. The Relation Similarity Service in Aqualog uses Query-Triples to create Onto-Triples where each term in the Query-Triples are matched to elements in the ontology by using string-based comparison methods and WordNet [14]. Evolving from AquaLog system, PowerAqua system [26] directs to the open-domain case by combining knowledge from various heterogeneous ontologies autonomously created on the Semantic web. Following Aqualog model, meanwhile, PANTO [61], relying on the statistical Stanford parser to produce a parse tree of an input natural language question, maps the input to query-triples. The query-triples are then translated into Onto-triples with the help of a lexicon of all entities extracted out of a given target ontology enlarged with WordNet synonyms. Finally, Onto-triples with potential words educed from the parse tree are used to produce SPARQL queries to the ontology.

Also, using the GATE framework, QuestIO [10] recognizes concepts inside an input question through gazetteers. Then QuestIO retrieves potential relations between identified concept pairs before ranking them due to the similarity, distance and specificity scores, and then dynamically creates formal queries such as SeRQL or SPARQL based on identified concepts and ranked relations. FREyA [11] is the successor to QuestIO, allowing users to enter questions in any form and involving the users to resolve ambiguities through a dialog if necessary. In ORAKEL [7], 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 good performance, it involves a customization process of domain-specific lexicon. Pythia [52] relies on the use of ontology-based grammars generated from Lexicalized Tree Adjoining Grammar tree to be able to process linguistically complex questions. However, Pythia requires a manually created lexicon. Another interesting work over linked data as detailed in [53] also proposed an approach to convert syntactic-semantic representations of input natural language questions to SPARQL templates.

2.4. *Question answering and question analysis for Vietnamese*

Turning to Vietnamese question answering, Nguyen and Le [33] introduced a NLIDB system in Vietnamese employing semantic grammars. Their system includes two main modules: QTRAN and TGEN. QTRAN (Query Translator) maps a natural language question to an SQL query while TGEN (Text Generator) generates answers based on the query result tables. QTRAN uses limited context-free grammars to analyze the user's question into a syntax tree via CYK algorithm [64]. The syntax tree is then converted into an SQL query by using a mapping dictionary to determine names of attributes in Vietnamese, names of attributes in the database and names of individuals stored in these attributes. TGEN module combines pattern-based and keyword-based approaches to make sense of the metadata and relations in the database tables in order to generate answers.

Our first KbQAS conference publication [34] reported a hard-wire approach to convert a Vietnamese natural language question into an intermediate representation element which is then used to extract the corresponding elements from the target ontology for returning the answer. Phan and Nguyen [43] later described a method to map Vietnamese questions into triple-like of *Subject*, *Verb* and *Object* in utilizing JAPE rules. Subsequently, Nguyen and Nguyen [35] presented another ontology-based QA System for Vietnamese, where keywords in a user's query are determined by using pre-defined templates and then producing SPARQL query to retrieve triple-based answer from ontology. In addition, Tran et al. [51] described the VPQA system to answer person name-related questions. Besides, Nguyen et al. [39] presented another Vietnamese NLIDB system, in economic-survey-data domain, using JAPE grammars for converting input questions into queries in R language to extract answers.

3. Our Question Answering System KbQAS

Figure 1 shows the architecture of our system which contains two components: the Natural language question analysis engine and the Answer retrieval.

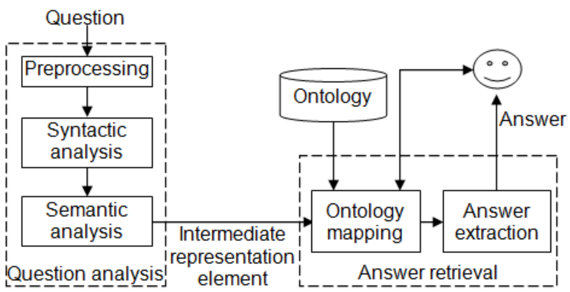


Figure 1. System architecture.

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

The answer retrieval component includes two main modules: Ontology Mapping and Answer Extraction. It takes an intermediate representation produced by the question analysis component and an Ontology as its input to generate semantic answers.

3.1. Intermediate Representation of an input question

Unlike Aqualog [25], the intermediate representation in our system is used to cover a wider variety of question types. It consists of a *question-structure* and one or more *query-tuple*s in the following format:

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

where $Term_1$ represents a concept (object class), $Term_2$ and $Term_3$, if exist, represent entities (objects or instances) excluding the cases of question-structures *Definition* and *Compare*. The *Relation* (property) is a semantic constraint among terms in the question.

This representation is aimed to capture the semantic of question. We define the following question-structures: *Normal, UnknTerm, UnknRel, Definition, Compare, ThreeTerm, Clause, Combine, And, Or, Affirm, Affirm_3Term, Affirm_MoreTuples* and question categories: *HowWhy, YesNo, What, When, Where, Who, Many, ManyClass, List* and *Entity* as described in **Appendixes A** and **B**, respectively.

Simple questions only have one *query-tuple* and its *question-structure* is the sub-structure of the tuple. Complex questions such as composite questions have several sub-questions, where each one is represented by a separate *tuple*, and the *question-structure* captures this composition attribute. Composite questions such as:

"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?

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

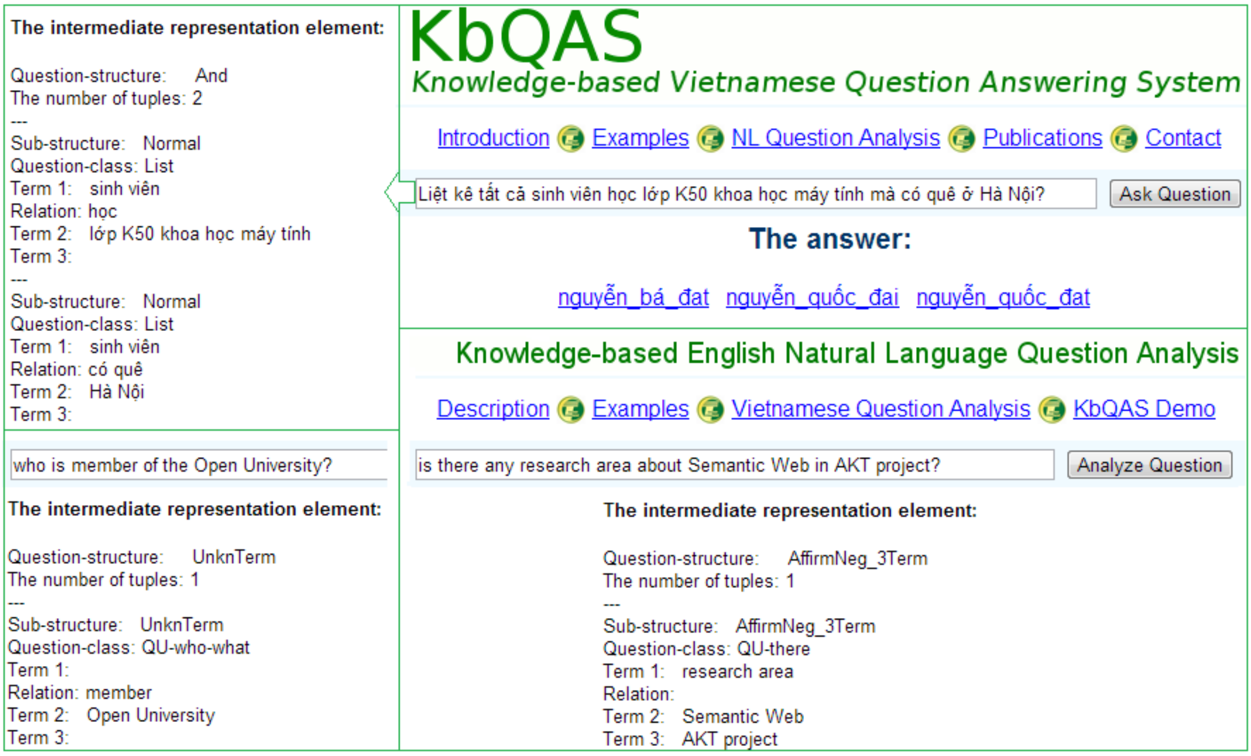


Figure 2. Illustrations of question analysis and question answering.

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

3.2. An illustrative example

For demonstration[1] and evaluation purposes, we reuse an ontology which models the organizational system of the University of Engineering and Technology, Vietnam National University, Hanoi [38]. The ontology contains 15 concepts like "trường$_{school}$", "giảng viên$_{lecturer}$", "sinh viên$_{student}$", 17 attributes or relations such as "học$_{study}$", "giảng dạy$_{teach}$", "là sinh viên của$_{is\ student\ of}$" and 78 instances as described in our first KbQAS version [34].

Given a complex-structure question:

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

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

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

The question analysis component determines that this question has question-structure *And* 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$_{Hanoi}$, ?)*.

Query-tuples are mapped to ontology-tuples by the Ontology mapping module in the Answer retrieval component: *(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$_{Hanoi}$)*. With each ontology-tuple, the Answer Extraction module finds all satisfied instances in the ontology, and then generates an answer based on the question-structure *And* and the question category *List*. Figure 2 shows the returned answer.

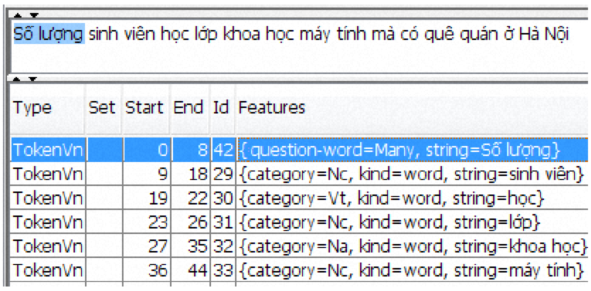
3.3. Natural language question analysis component

Natural language question analysis component is the first component in any QA system. When a question is asked, the task of the component is to translate the natural language question to an intermediate representation of the input question, which is utilized in the rest of the system.

KbQAS makes the use of the JAPE grammars in the GATE framework [9] to specify regular expression patterns based on semantic annotations for question analysis, in which existing linguistic processing modules for Vietnamese including Word Segmentation, Part-of-speech tagging [41] are wrapped as GATE plug-ins. The results of the modules are annotations capturing information such as sentences, words, nouns and verbs. 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. New modules of preprocessing, syntactic analysis, and semantic analysis 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 as part-of-speech as displayed in figure 3. Vietnamese is a monosyllabic language; hence, a word may contain more than one token. However, the Vietnamese word segmentation module is not trained for question domain. 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*$_{when}$*"*, *"là cái gì*$_{what}$*"*. In this module, we identify those and mark them as single *TokenVn* annotations with corresponding *"question-word"* feature and its semantic categories like $HowWhy_{cause\ |\ method}$, $YesNo_{true\ or\ false}$, $What_{something}$, $When_{time\ |\ date}$, $Where_{location}$, $Many_{number}$, Who_{person}. In fact, this information will be used in creating rules in the syntactic analysis module at a later stage.



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.

In addition, we marked phrases that refer to comparative phrases (such as *"lớn hơn*$_{greater\ than}$*" "nhỏ hơn hoặc bằng*$_{less\ than\ or\ equal\ to}$*"* ...) or special-words (for example, abbreviation of some words on special-domain) by single *TokenVn* annotations.

3.3.2. Syntactic analysis

This module is responsible for identifying noun phrases and the relations between noun phrases. The different modules communicate through the annotations, for instance, this module uses the *TokenVn* annotations which are the output of the previous 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 important that we can reliably detect noun phrases in order to generate the *query-tuple*. Based on the grammar of Vietnamese language [12], we use the *JAPE* grammars to specify patterns over annotations as shown in Table 1. When a noun phrase is matched, an annotation *NounPhrase* is created to mark up the noun phrase. Moreover, its *"type"* feature is used to identify the concept and entity that are contained in the noun phrase using the following heuristic:

If the noun phrase contains a single noun (not including numeral nouns) and does not contain a proper noun, it contains a *concept*. If the noun phrase contains a proper noun or contains at least three single nouns, it contains an *entity*. Otherwise, concepts and entities are determined using a manual dictionary. In this step, a manual dictionary is built for describing concepts and their corresponding synonyms in the Ontology.

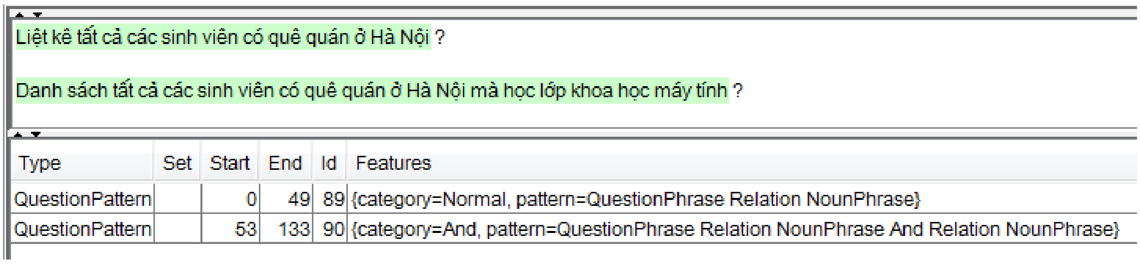


Figure 4. Examples of question-structure patterns.

In addition, the question-phrases are detected by using noun phrases and question-words identified by the preprocessing module. *QuestionPhrase* annotations are generated to cover question-phrases with a corresponding *"category"* feature which gives information about question categories.

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

(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 following question as referred to 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 phrase 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

This module aims to identify the question-structure and produce the query-tuples as the intermediate representation (*sub-structure, question-category, Term$_1$, Relation, Term$_2$, Term$_3$)* of the input question using the annotations generated by the previous modules. Existing *NounPhrase* annotations, and *Relation* annotations are potential candidates for terms and relations respectively, while *QuestionPhrase* annotations covering matched question-phrases are used to detect the question-category.

In the first KbQAS version [34], following Aqualog [25], we developed an ad-hoc approach to detect question patterns and then use the patterns for creating the intermediate representation. For instance, figure 4 presents the detected structure patterns of the 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 them by using annotations returned by pre-processing 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 of input question is created in a hard-wire manner linking every detected pattern via JAPE grammars to extract corresponding elements. This hard-wire manner takes a lot of time and effort to handle new patterns. For example, as can be seen in figure 4, the hard-wire approach is unable to reuse the detected structure pattern of the first question for identifying the pattern of the second one. As rules are created in an ad-hoc manner, the hard-wire one 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. This is considered as the key innovation of our KbQAS system.

3.4. Answer retrieval component

The Answer retrieval component includes two main modules: Ontology Mapping and Answer Extraction as shown in figure 1. It takes an intermediate representation produced by the question analysis component and an Ontology as its input to generate an answer. We employed the Relation similarity service component of the AquaLog system [25] to develop the Answer retrieval component.

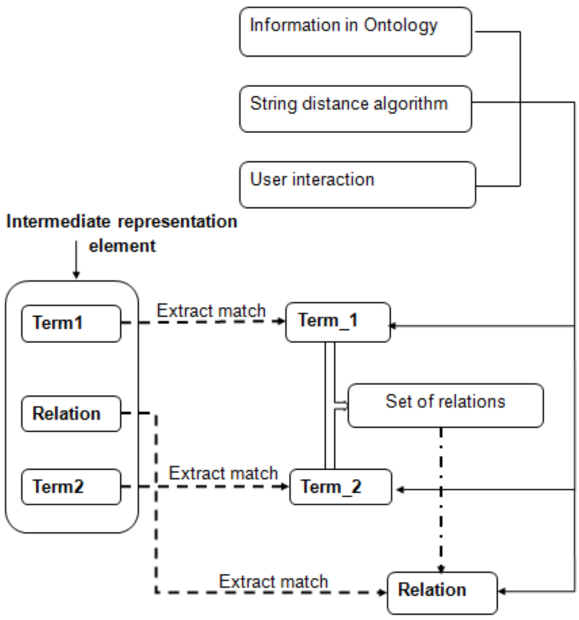


Figure 5. Mapping Ontology module for query-tuple with two terms and one relation.

The task of the Ontology Mapping module is to map terms and relations in the query-tuple to concepts, instances and relations in the Ontology by using string names. If an exact match is not possible, we employ a manually built lexicon of synonyms and a string distance algorithm as presented in [55] to find near-matched elements in the Ontology with the similarity measure above a certain threshold. In case ambiguity is still present, the KbQAS system interacts with the users by presenting different options to get the correct ontology element. For instance, with 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 returns query-tuple *(Normal, List, sinh viên*$_{student}$*, học*$_{study}$*, lớp khoa học máy tính*$_{computer\ science\ course}$*, ?)*. As the Ontology Mapping cannot find the exact instance corresponding with *"lớp khoa học máy tính*$_{computer\ science\ course}$*"* in the target ontology, it requires users 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*$_{department}$*"* in the ontology.

Following the AquaLog, for each query-tuple, the result of the Mapping Ontology module is an ontology-tuple where the terms and relations in the query-tuple are now their corresponding elements in the Ontology. How the Mapping Ontology module finds corresponding elements in the 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. It first tries to match $Term_1$ and $Term_2$ with concepts or instances in the Ontology. After that, the set of potential relations in the Ontology contains only relations between the two mapped concepts/instances. The ontology relation is then identified in a similar manner as a mapping term to a concept or an instance.

With the ontology-tuple, the Answer Extraction module finds all individuals of the corresponding Ontology concept of $Term_1$, having the ontology relation with the individual corresponding to $Term_2$. Depending on the question-structure and question category, the answer will be returned.

4. Single Classification Ripple Down Rules for Question Analysis

As mentioned in section 3.3.3, because of the complexity of the representation and the variety of question-structure types, manually creating the rules in an ad-hoc manner is very expensive and error-prone. For example, many rule-based approaches as indicated in [25,34,43] manually defined a list of sequence pattern structures to analyze questions. As rules are created in an ad-hoc manner, these approaches share common difficulties in managing the interaction between rules and keeping consistency among them.

In this section, we will introduce our knowledge acquisition approach[2] to analyze natural language questions by applying SCRDR methodology [8,45] 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 way that interaction among rules is controlled and consistency is maintained.

A SCRDR knowledge base is built to identify the question-structure and to produce the query-tuples as the intermediate representation. We will firstly outline the SCRDR methodology and propose a rule language for extracting this intermediate representation for a given input 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 Ripple-Down Rules [8,45] which inspired our knowledge acquisition approach for question analysis. Ripple Down Rules methodology allows one to add rules to a knowledge base incrementally without the need of a knowledge engineer. A new rule (i.e. an exception rule) is only created when the knowledge base performs unsatisfactorily on a given case. The rule represents an explanation for why the conclusion should be different from the knowledge base's conclusion on the case at hand.

A Single Classification Ripple Down Rules (SCRDR) [8,45] tree as illustrated in figure 6 is a binary tree with two distinct types of edges. These edges are typically called *except* and *false* edges (or can be named *except* and *if-not* 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 the *conclusion*.

Cases in SCRDR are evaluated by passing a case to the root of the tree. At any node in the tree, if the condition of a node N's rule is satisfied with the case, the case is passed to the exception child of node N using the *except* link if it exists. Otherwise, the case is passed on to the node N's *false* child. The conclusion given by this process is the conclusion from the last node in the tree which *fired* (satisfied by the case).

[2]Vietnamese question analysis demonstration is available on-line at http://150.65.242.39:8080/KbVnQA/
English question analysis demonstration is available on-line at http://150.65.242.39:8080/KbEnQA/

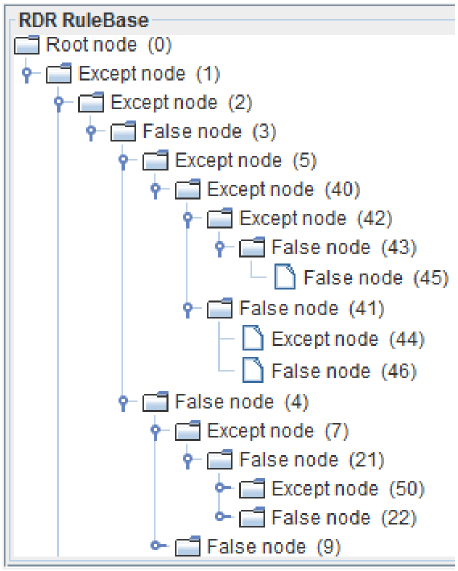


Figure 6. A part of our SCRDR tree for processing English questions.

To ensure that a conclusion is always given, the root node typically contains a *trivial* condition which is always satisfied. This node is called the *default* node. For instance, the root node in our SCRDR knowledge base constructed for analyzing English questions as shown in figure 6 corresponds with a default rule of "*if* **True** *then* **null**". It means that every case (i.e. question) will be satisfied by the condition of the default rule at the root node, however, the rule gives a *null* conclusion (i.e. an empty intermediate representation element for the question). The default rule is the unique rule which is not an exception rule of any other rule.

Starting with an empty SCRDR knowledge base consisting of only default node, the process of building the knowledge base can be performed automatically [37], or manually [42,36]. A new node containing a new rule (i.e. a new exception rule) is added to an SCRDR tree when the evaluation process returns the *wrong* conclusion. The new node is attached to the last node in the evaluation path of the given case with the *except* link if the last node is the *fired* one. Otherwise, it is attached with the *false* link.

Section 4.3 will demonstrate the construction process of the SCRDR tree displayed in figure 6. With the tree, the rule at node (1) (simply, rule 1) is the exception rule of the default rule (rule 0). Rule 2 is an exception rule of rule 1. As node (3) is the false-child node of node (2), the associated rule 3 is also an exception rule of rule 1. Furthermore, both rules 4 and 9 are also exception rules of rule 1. Similarly, rules 40 and 41 are exception rules of rule 5 whereas rules 42, 43 and 45 are all 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 of exception structure.

Given the case "who are the partners involved in AKT project?", it is satisfied by the default rule at root node, it then is passed to the node (1) using except link. As the case satisfies the condition of rule 1, it is passed to the node (2). Because it does not satisfy the condition of the rule 2, it will be transferred to the node (3) using false link. Since the case satisfies conditions of rules 3, 5 and 40 and does not fulfill conditions of rules 42, 43 and 45, we have the evaluation path (0)-(1)-(2)-(3)-(5)-(40)-(42)-(43)-(45) with fired node at (40). With the case of "in which projects is enrico motta working on", it satisfies conditions of rules 0, 1 and 2. As node (2) has no except-child node, we have evaluation path (0)-(1)-(2) and fired node at (2).

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 [9]. 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 annotations in the pattern by requiring 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 --> :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 catch phrases starting with a *TokenVn* annotation covering either the word *"liệt kê$_{list}$"* or the word *"chỉ ra$_{show}$"*, followed by a *concept*-typed noun phrase surrounded by a *NounPhrase* annotation. When applying the pattern on a text fragment, *QuestionPhrase* annotations having *"category"* feature with its value of *"List"* would be posted over phrases matching the pattern.

Additionally, the condition part of the rule can include additional constrains. For example, in rule 40 in figure 6, the addition constrain *"RDR1_QP.hasanno == QuestionPhrase.category == QU-whichClass"* is used to make a prerequisite condition, which requires a RDR1_QP annotation that must have a QuestionPhrase annotation covering their substring with *"QU-whichClass"* as the value of its *"category"* feature.

The rule's conclusion contains the question-structure and the query-tuples corresponding to the intermediate representation where each element in the query-tuple is specified by a newly posted annotations 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 prefix RDR 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 the next sub-section of knowledge acquisition process.

Given a new input question, a rule's condition is considered satisfied if the whole input question is matched by the condition pattern. The conclusion of the fired rule outputs the intermediate representation of the input question. To create rules for capturing structures of questions, we use patterns over annotations returned by the previous modules of preprocessing and syntactic analysis.

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, therefore, it is language independent[3]. The language-specific part is in the rules itself. Consequently, in this section, we illustrate the process building the SCRDR knowledge base of rules as presented in figure 6 for processing English natural language questions. We utilized JAPE grammars employed in AquaLog [25] for detecting the prepositions, noun phrases, question phrases, and relation phrases in English questions. As our Vietnamese question-category definitions is not suitable to adapt to the English domain, we reused those of the AquaLog system. Figure 7 shows the graphic user interface of our natural language question analyzer.

• The rest of this section describes how the knowledge base building process works. In contrast to the

[3]The illustration of building a knowledge base of rules for analyzing Vietnamese questions can be found in our conference publication [36].

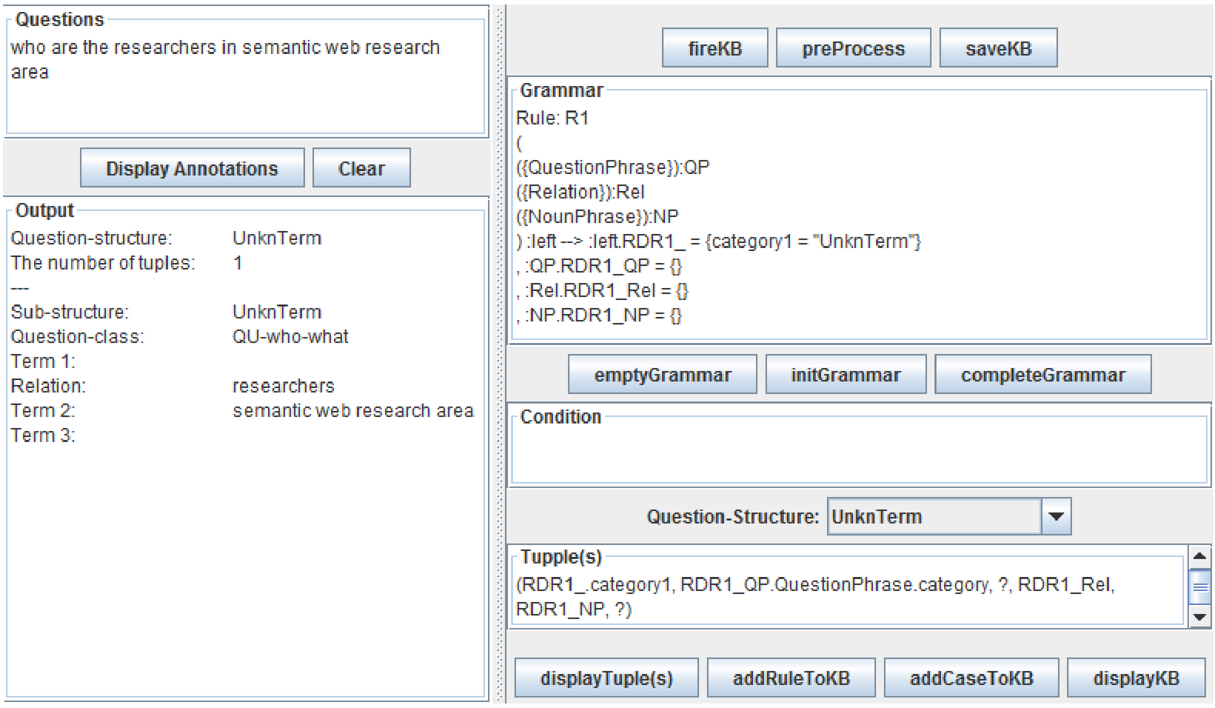


Figure 7. Question analysis module to create the intermediate representation of question *"who are the researchers in semantic web research area?"*

.

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

When we encountered the question: *"who are the researchers in semantic web research area ?"*

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

Supposed we start with an empty knowledge base, the fired rule is the default rule **R0** that gives an empty conclusion - an incorrect intermediate representation. This can be corrected by adding the following rule as an exception rule of the rule **R0** to the knowledge base:

Rule: R1

(
({QuestionPhrase}):QP
({Relation}):Rel
({NounPhrase}):NP
) :left $\dashrightarrow$:left.RDR1_ = {category1 = "UnknTerm"}
, :QP.RDR1_QP = {}
, :Rel.RDR1_Rel = {}
, :NP.RDR1_NP = {}

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

If the condition of rule R1 matches whole input question, a new annotation RDR1_ will be created to entirely cover the input question and new annotations RDR1_QP, RDR1_Rel and RDR1_NP will also be generated for covering sub-phrases of the input question. Once rule **R1** is fired, the matched input question is deemed to have a query-tuple with *sub-structure* taking the value of the *"category1"* feature of RDR1_ annotation, *question-category* taking the value of the *"category"* feature of QuestionPhrase annotation surrounding the same span as RDR1_QP annotation. In addition, the query-tuple's *Relation* is the string covered by RDR1_Rel, $Term_2$ is the string surrounded by RDR1_NP while $Term_1$ and $Term_3$ are missing. The example of firing at rule **R1** is displayed in figure 7.

Assumed that, in addition to the default rule **R0** and rule **R1**, the current knowledge base contains rule **R2** as an exception rule of the rule **R1**. In the SCRDR tree structure, the associated node (2) is the except-child node of the node (1) as displayed in the figure 6. When we encountered 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 evaluation path of (0)-(1)-(2) with the fired rule **R1**. However, rule **R1** produces an incorrect conclusion of question-structure *UnknRel* and query-tuple *(UnknTerm, QU-whichClass, ?, ?, Knowledge Media Institute, ?)* as the RDR1_ annotation only covers a part of the question and *"are"* is not considered as a relation. The following rule **R3** would be appended as an exception rule of the fired rule **R1** to knowledge base:

Rule: R3

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

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

In the SCRDR tree structure, the corresponding node (3) is added as the false-child node of the 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 fired rule **R3**. Using rule **R3**, the correct output of the input question is question-structure *Normal* and query-tuple *(Normal, QU-whichClass, universities, collaborating, Knowledge Media Institute, ?).*

Similarly, another input question made an attachment of the rule **R4** which is an exception rule of the rule **R1**. In the SCRDR tree structure, the associated node (4) is linked to node (3) by *false* edge.

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 evaluation path of (0)-(1)-(2)-(3) and rule **R3** is the fired rule. But rule **R3** returns a wrong conclusion. The following rule **R5** is added to correct the conclusion as an exception rule of the rule **R3** :

Rule: R5

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

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

As the node (3) is the last node in the evaluation path, the corresponding node (5) is attached as the except-child node of the node (3) as displayed in figure 6. Using the rule **R5**, we have the correct conclusion consisting of question-structure *Normal* and query-tuple (*Normal, QU-who-what, partners, involved, AKT project, ?*).

• The processes of adding above rules illustrate the ability of quickly handling new question-structure patterns of our knowledge acquisition approach against the hard-wire manners [25,34]. The following examples demonstrate the ability of our method in solving 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]]

it will be fired at node (5) which is the last node in the evaluation path of (0)-(1)-(2)-(3)-(5). But rule **R5** gives the wrong conclusion of question-structure *Normal* and query-tuple (*Normal, QU-whichClass, publications, related to, semantic portals, ?*). We add node (40) containing the following exception rule **R40** as the except-child node of the node (5) to correct the conclusion returned by the rule **R5** in using constrains via rule condition:

Rule: R40

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

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

Conclusion: question-structure *Clause*[4] 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, ?*).

[4] Clause question-structure is defined as consisting of two query-tuples that returned results of sub-question represented by second query-tuple indicate missing element of $Term_2$ in the first query-tuple. The readers can find more details in our question-structure definitions in the appendix A.

The additional condition of rule **R40** matches a RDR1_QP annotation that has a QuestionPhrase annotation covering their substring with *QU-whichClass* as the value of its *"category"* feature. The extra annotation constraint of *hasAnno* requires that the text covered by the annotation must contain the specified annotation. Additionally, the values of features *"category1"* and *"category2"* of *RDR40_* annotation are assigned to the corresponding query-tuples' *substructure*. Rule **R40** generates the correct output of question-structure *Clause* and query-tuples (*Normal, QU-whichClass, researchers, wrote, ?, ?*) and (*Normal, QU-whichClass, publications, related to, semantic portals, ?*).

When it came to another question:

"which projects sponsored by eprsc are related to semantic web ?"

[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]]

The current knowledge base generates an evaluation path of (0)-(1)-(2)-(3)-(5)-(40)-(42)-(43) with the fired rule **R40**. However, rule **R40** returns a wrong conclusion with question-structure *Clause* 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 an exception rule the rule **R40** is added to correct the conclusion given by the rule **R40**:

Rule: R45

(

{RDR40_}

) :left --→ :left.RDR45_ = {category1 ="Normal", category2 = "Normal"}

Condition: RDR1_Rel.hasanno == Token.category == VBN[5]

Conclusion: question-structure *And* and two query-tuples of

(*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, ?*).

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

[5]*Token* annotations are generated as outputs of the English tokenizer, sentence splitter and POS tagger in GATE framework [9].

5. Experiments

We separately evaluate the question analysis and answer retrieval components in section 5.1 and section 5.2, respectively. The reason is that the method employed in the question analysis component is domain-and-language independent while the answer retrieval component is to extract the answers from a domain-specific ontology.

5.1. Experiments on analyzing questions

This section is to indicate the ability of using our question analysis approach for quickly building a new knowledge base, and then for easily adapting to a new domain and a new language. We evaluate both our approaches of hard-wire manner (section 3.3.3) and knowledge acquisition (section 4) on Vietnamese question analysis, and later present the experiment in building a SCRDR knowledge base for processing English questions using the same intermediate representation.

5.1.1. Question Analysis for Vietnamese

Based on a training set of 400 various-structure questions generated by four volunteer students, we build a knowledge base of 92 rules. We evaluate the quality of the knowledge base on an unseen list of 88 questions related to the VNU University of Engineering and Technology. Table 2 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 in layer 0. This indicates that the exception structure is indeed present and even extends to level 4.

Table 2

Number of exception rules in layers in our SCRDR KB

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

In our experiment, we evaluate both our approaches to analyzing questions including the first one of hard-wire manner as mentioned in the section 3.3.3 and the second of knowledge acquisition for building SCRDR knowledge base, using the same training set of 400 questions and test set of 88 questions. Our second method took one expert about 13 hours to build a KB. However, most of the time was spent in looking at questions to determine if they belong to the structure of interest and which phrases in the sentence need to be extracted for the intermediate representation. The actual time required to create 92 rules by one expert is only about 5 hours in total. In contrast, implementing question analysis component corresponding to our first method took about 75 hours for creating rules in an ad-hoc manner. Anecdotal account indicates that the cognitive load in creating rules in the second approach is much less compared to that in the first one as in our case, we do not have to consider other rules when crafting a new rule.

Table 3
Number of correctly analyzed questions

Type	#questions
Our first approach driving hard-wire manner	70 (79.5%)
Our second approach of knowledge acquisition	74 (**84.1%**)

Table 3 shows the number of correctly analyzed questions of our approaches. By using knowledge base for resolving some ambiguous cases, the second approach accounting for 74 of 88 questions to obtain the accuracy of 84.1% performs better than the first one. Table 4 provides the sources of errors for the remaining questions that our second approach incorrectly processes. It points out that most errors come from unexpected structures. This could be easily rectified by adding more exception rules to the current knowledge base, especially when we have a bigger training set that contain a larger variety of question-structure types.

Table 4
Number of incorrectly analyzed questions

Reason	#questions
Unknown structures of questions	12
Word segmentation was not trained for question-domain	2

For instance of failure by word segmentation without training over questions domain, given 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?"*), the output of existing linguistic processing modules for Vietnamese [41] wrapped as GATE plugins [9], that we used, assigns the word "quê$_{hometown}$" as an adjective instead of a noun. Thus, "quê$_{hometown}$" is not covered by NounPhrase annotation leading the unknown structure pattern of the question.

Table 5
Number of rules corresponding to each question-structure type (QST) in the knowledge bases for Vietnamese (#V) and English (#E), and the number of Vietnamese testing questions (#TQ) and the number of Vietnamese correctly answered questions (#CA) associating to each rule.

QST	#V	#CA	#TQ	#E
Definition	2	1	2/2	3
UnknRel	4	4	4/7	4
UnknTerm	7	6	7/7	3
Normal	7	7	7/7	8
Affirm	10	5	5/5	4
Compare	5	0	2/4	0
ThreeTerm	9	7	7/10	5
Affirm_3Term	5	4	4/4	3
And	9	7	8/8	14
Or	23	18	21/24	1
Affirm_MoreTuples	3	1	2/3	0
Clause	6	0	4/5	13
Combine	1	1	1/2	0
Total:	91	61	74/88	58

Regarding to question-structure-based evaluation, table 5 presents the number of rules built in the Vietnamese knowledge base in corresponding for each type of question-structure and the number of corresponding testing questions associated with each rule. For example, in the second row and fourth column of table 5, with 7 testing questions tending to have question-structure *UnknRel*, there are 4 testing questions correctly analyzed, and remaining 3 testing questions incorrectly processed.

5.1.2. *Question Analysis for English*

For the experiment in English, we take 170 English question examples of AquaLog[6] [25], which Aqualog is able to correctly analyze. Those questions are specified to the Knowledge Media Institute and its research area on semantic web. Using our knowledge acquisition approach, we built a knowledge base of 59 rules including the default one. It took 7 hours to build the knowledge base, which includes 3 hours of actual time to create all rules. Table 6 shows the number of rules in English knowledge base layers while the number of

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

rules corresponding with each question-structure type is presented table 5.

Table 6
Number of exception rules in layers in our English SCRDR KB

Layer	Number of rules
1	9
2	13
3	20
4	11
5	5

In order to demonstrate that our approach could be applied to an open domain, we use the built English knowledge base to process a test set of 500 questions[7] from the TREC-10 Question Answering Track [57].

Table 7
Number of questions corresponding with each question-structure type

Question-structure type	#questions
Definition	130
UnknTerm	66
UnknRel	4
Normal	20
ThreeTerm	15
And	6

Table 7 presents the number of correctly analyzed questions corresponding with each question-structure type. Table 8 gives the sources of errors for 259 incorrect cases. This could be corrected by adding more exception rules to the current English knowledge base in the use of a larger training data set such as the corpus of 5500 open domain questions[8] [22].

Table 8
Error results

Reason	#questions
Have special characters (such as **/ – “ ” ’s**) and abbreviations	64
Not have compatible patterns	185
Semantic error in elements of the intermediate representation	10

As the intermediate representation of our system is different to AquaLog, it is impossible to directly compare our approach with Aqualog on the English domain. However, the experiments are indicative of the ability in using our approach to quickly build a new knowledge base for a new domain and a new language.

5.2. Experiment on answering Vietnamese questions

To evaluate our KbQAS system by specifying in the Answer retrieval component, the ontology modeling the organizational structure of the VNU University of Engineering and Technology as mentioned in the section 3.2 is used as target domain. This ontology was manually constructed by using the Protégé platform [16]. From the list of 88 questions as mentioned in section 5.1.1, we employed 74 questions for which our question analysis component correctly processed.

Table 9
Questions successfully answered

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

The performance result is shown in table 9. The answer retrieval component gives correct answers to 61 questions (over 74 questions) to obtain a promising accuracy of 82.4%. Out of those, 30 questions can be answered automatically without interaction with the user. The number of correctly answered questions corresponding with each question-structure type can be found in the third column of table 5.

Table 10
Questions with unsuccessful answers

Type	# questions
Ontology mapping errors	6
Answer extraction errors	7

Table 10 gives the limitations that would have to be handled in future KbQAS versions. The errors occurred by the Ontology mapping module are because of the ontology construction of lacking domain-specific conceptual coverage and the few relationships between concept pairs. This leads to that specific terms or relations in the intermediate representation cannot be mapped or incorrectly mapped to corresponding elements in the target ontology to produce the Ontology-tuple. Furthermore, KbQAS fails to extract the answers to 7 other questions due to: (i) Dealing with questions belonging to structures of “Compare” involves specific services. For example, handling the question

[7] http://cogcomp.cs.illinois.edu/Data/QA/QC/TREC_10.label
[8] http://cogcomp.cs.illinois.edu/Data/QA/QC/train_5500.label

"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 for ranking students according to their GPA. (ii) In terms of "Clause" questions and one "Affirm_MoreTuples" question, combining sub-questions triggers complex inference tasks and bugs which cannot be resolved in this version.

6. Conclusion

In this paper, we described the ontology-based Vietnamese question answering system KbQAS. It consists of two components of the Natural language question analysis engine and the Answer retrieval. The two components are connected by an intermediate representation element capturing the semantic structure of the input question, facilitating the processing of matching with the target ontology to find the answer. To the best of our knowledge, this is the first ontology-based question answering system for Vietnamese.

Additionally, we proposed our knowledge acquisition approach to systematically acquiring rules for converting a natural language question into an intermediate representation. Given a complex intermediate representation of a question, our approach allows systematic control of interactions between rules and keeping consistency among them. We believe our knowledge acquisition approach for question analysis 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 using our system. Incrementally, an annotated corpus and a rule-based system can be obtained simultaneously. The structured data used in the question analysis evaluation falls into the category of querying database or ontology but the problem of question analysis we tackle go beyond that, as it is a process that happens before the querying process. It can be applied to question answering in open domain against text corpora as long as the technique requires an analysis to turn the input question to an explicit representation of some sort.

Experimental results of our KbQAS system with a wide range of questions are promising. Specifically, the answer retrieval module achieves an accuracy of 82.4%. Moreover, the experiments - on analyzing natural language questions with an accuracy of 84.1% for the Vietnamese corpus and time of 7 hours to build the English knowledge base - show that our knowledge acquisition approach for question analysis enables ones to easily build a new system or adapt an existing system to a new domain or a new language. In the future, we will extend our system to employ a near match mechanism to improve the generalization capability of existing rules in the knowledge base and to assist the rule creation process.

Appendix

A. Definitions of question-structures

We define question-structures: *Normal, UnknTerm, UnknRel, Definition, Compare, ThreeTerm, Clause, Combine, And, Or, Affirm, Affirm_MoreTuples, Affirm_3Term* as below. In each query-tuple, in general, $Term_1$ represents a concept, excluding cases of *Affirm*, *Affirm_3Term* and *Affirm_MoreTuples*.

- A question will have question-structure *Normal* if it has only one query-tuple, and the query-tuple's $Term_3$ is missing.
- A question will have question-structure *UnknTerm* if it has only one query-tuple, and the query-tuple lacks $Term_1$ and $Term_3$.
- A question will have question-structure *UnknRel* if it has only one query-tuple in the lack of *Relation* and $Term_3$. For instance, the question "List all the publications in knowledge media institute" has question-structure *UnknRel* and query-tuple *(UnknRel, QU-listClass, publications, ?, knowledge media institute, ?)*.
- A question will have question-structure *Definition* if it has only one query-tuple lacking all of $Term_1$, *Relation* and $Term_3$. For instance, the question "what are research areas?" has a query-tuple *(Definition, QU-who-what, ?, ?, research areas, ?)*.
- If a question belongs to one of three question-structure types *Normal*, *UnknRel* and *UnknTerm* and appears in question category *YesNo*, the question will have question-structure *Affirm*. For example, the question "Is Tran Binh Giang a Phd student?" has question-structure *Affirm* and query-tuple *(Affirm, YesNo, Phd student, ?, Tran Binh Giang, ?)*
- A question will have question-structure *ThreeTerm* if it has only one query-tuple, and it allows to miss either $Term_1$ or *Relation*. For instance, the question "Who is the director of the compendium project in Knowledge Media?" has question-structure *ThreeTerm*

and query-tuple *(ThreeTerm, QU-who-what, ?, director, compendium project, Knowledge Media)*.

• If a question has question-structure *ThreeTerm* and also belongs to *YesNo* category, it will have question-structure *Affirm_3Term*. Given 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?"*), it has query-tuple (*Affirm_3Term, ManyClass, sinh viên$_{student}$, học$_{study}$, lớp K50 khoa học máy tính$_{K50computersciencecourse}$, 45*). Another example for this type of question-structure is illustrated in figure 2.

• A question will have question-structure *Compare* if it belongs to one of three question-structure types *Normal*, *UnknRel* and *UnknTerm*, and contains a comparing-phrase which is detected by preprocessing module; the query-tuple's *Term*$_3$ in this case is used to hold this 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 query-structure of *Compare* and query-tuple (*Normal, Entity, sinh viên$_{student}$, điểm trung bình$_{grade\ point\ average}$, khoa công nghệ thông tin$_{faculty\ of\ Information\ Technology}$, cao nhất$_{highest}$*).

• If a question contains either token *"mà$_{and}$"/"và$_{and}$"* or *"hoặc$_{or}$"*, it will have two or more query-tuples corresponding with *And* or *Or* question-structure respectively. For instance, the question "which projects are about ontologies and the semantic web?" has question-structure *And* 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 question-structure *And* 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 question-structure *Or* and two query-tuples *(UnknTerm, QU-who-what, ?, interested, ontologies, ?)* and *(UnknTerm, QU-who-what, ?, interested, semantic web, ?)*.

However, with some question as *"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?"*), it contains *"và$_{and}$"*, but it has question-structure *Or* 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$_{Pham\ Duc\ Dang}$, ?*).

• If a question appearing in question category *YesNo* belongs to question-structure types *And* or *Or*, it will have question-structure *Affirm_MoreTuples*. 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 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$_{faculty\ of\ Mathematics}$, ?)*.

• If a question associates with two query-tuples and returned results of one query-tuple is considered as miss-element in the remaining query-tuple, it will have question-structure *Clause*.

For example, the question "how many projects are headed by researchers in the open university?" has question-structure *Clause* and query-tuples *(Normal, QU-howmany, projects, headed, ?, ?)* and *(UnknRel, QU-howmany, researchers, ?, open university, ?)*.

Specially, in case of the question contains comparing-phrase like *"số lượng sinh viên học lớp K50 khoa học máy tính lớn hơn 45 phải không ?"*[9] (the number of students studying in K50 computer science course is higher than 45, is not it?) will have question-structure *Clause* 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$_{K50\ computer\ science\ course}$, ?)*.

• If a composite question is constructed from two or more independent sub-questions, it will have question-structure *Combine*. 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 two query-tuples (*UnknTerm, Who, ?, có quê quán$_{has\ hometown}$, Hà Tây$_{Hatay}$, ?*) and (*UnknTerm, Who, ?, học$_{study}$, khoa công nghệ thông tin$_{faculty\ ofInformation\ Technology}$, ?*).

B. Definitions of Vietnamese question-categories

In our system, 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

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

JAPE grammars using *NounPhrase* annotations and the *question-word* information identified by the preprocessing module. Obviously using this method in question-phrases detection phase will result in ambiguity when a question belongs to multiple categories.

• A *HowWhy*-type question refers causes or methods by containing the word re-annotated by single *TokenVn* annotation such as *"tại sao$_{why}$"*, or *"vì sao$_{why}$"*, or *"thế nào$_{how}$"*, or *"là như thế nào$_{how}$"*. In English, it is similar to *Why*-question or *How is/are* question.

• A *YesNo*-type question requires true or false answer by holding the word re-covered by single *TokenVn* annotations such as *"có đúng là$_{is\ that}$"*, or *"có phải là$_{is\ this}$"*, or *"phải không$_{are\ there}$"*, or *"đúng không$_{are\ those}$"*.

• A *What*-classified question refers to something in consisting of the word *"cái gì$_{what}$"*, or *"là gì$_{what}$"*, or *"là những cái gì$_{what}$"*. In English, this question type is *What is/are*-question-like.

• A *When*-type question contains the word relabelled by single *TokenVn* annotation such as *"khi nào$_{when}$"*, or *"vào thời gian nào$_{which\ time}$"*, or *"lúc nào$_{when}$"*, or *"ngày nào$_{which\ date}$"*.

• A *Where*-classified question requires answers about location in containing words such as *"ở nơi nào$_{where}$"*, or *"là ở nơi đâu$_{where}$"*, or *"ở chỗ nào$_{where}$"*.

• A question will be categorized to *Who* class, if it contains the word indicating answer referring to a person such as *"là những ai$_{who}$"*, or *"là người nào$_{who}$"*, or *"những ai$_{who}$"*.

• A question expecting the answer about number will belong to *Many* class (in English, these questions are *How much/many is/are*-questions). This question type holds the word like *"bao nhiêu$_{how\ much|many}$"*, or *"là bao nhiêu$_{how\ much|many}$"*, or *"số lượng$_{how\ many}$"*.

• A question will appear in *ManyClass* class, if it contains the word like *"bao nhiêu$_{Howmany}$"*, or *"số lượng$_{Numberof}$"* ... followed by a noun phrase (in English, this type is the same kind of *How many NounPhrase*-question).

• A question will appertain in *Entity* category if it holds a noun phrase followed by the word *"nào$_{which}$"* or *"gì$_{what}$"* (in English, this kind of question belongs to set of *which/what Noun Phrase* questions such as: which students, what class,...).

• A question will categorized to *List* class if it contains the word referring commands such as: *"cho biết$_{give}$"*, *"chỉ ra$_{show}$"*, *"kể ra$_{tell}$"*, *"tìm$_{find}$"*, *"liệt kê$_{list}$"* ... followed by a noun phrase.

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] Ion Androutsopoulos, Graeme Ritchie, and Peter Thanisch. Natural language interfaces to databases - an introduction. *Natural Language Engineering*, 1(1): 29–81, 1995.

[3] Paolo Atzeni, Roberto Basili, Dorte Haltrup Hansen, Paolo Missier, Patrizia Paggio, Maria Teresa Pazienza, and Fabio Massimo 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, NLDB 2004*, pages 413–420, 2004.

[4] Roberto Basili, Dorte H. Hansen, Patrizia Paggio, Maria Teresa Pazienza, and Fabio Massimo Zanzotto. Ontological resources and question answering. In *HLT-NAACL 2004: Workshop on Pragmatics of Question Answering*, pages 78–84, 2004.

[5] Delphine Bernhard and Iryna Gurevych. Answering Learners ' Questions by Retrieving Question Paraphrases from Social Q & A Sites. In *Proceedings of the Third Workshop on Innovative Use of NLP for Building Educational Applications*, EANL '08, pages 44–52, 2008.

[6] Robin D. Burke, Kristian J. Hammond, Vladimir A. Kulyukin, Steven L. Lytinen, Noriko Tomuro, and Scott Schoenberg. Question answering from frequently asked question files: Experiences with the faq finder system. *AI Magazine*, 18(2):57–66, 1997.

[7] Philipp Cimiano, Peter Haase, Jörg Heizmann, Matthias Mantel, and Rudi Studer. Towards portable natural language interfaces to knowledge bases - the case of the orakel system. *Data & Knowledge Engineering*, 65(2):325–354, 2008.

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

[9] Hammish Cunningham, Diana Maynard, Kalina Bontcheva, and Valentin 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, ACL'02*, pages 168–175, 2002.

[10] Danica Damljanovic, Valentin Tablan, and Kalina Bontcheva. A text-based query interface to owl ontologies. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*, 2008.

[11] Danica Damljanovic, Milan Agatonovic, and Hamish 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*, ESWC'10, pages 106–120, 2010.

[12] Quang Ban Diep. *Ngữ pháp tiếng Việt (Grammar of Vietnamese language)*. Vietnam Education Publishing House, 2005.

[13] Tiansi Dong, Ulrich Furbach, Ingo Glöckner, and Björn 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 - Volume Three*, IJCAI'11, pages 2430–2435, 2011.

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

[15] Ulrich Furbach, I Glöckner, and B Pelzer. An application of automated reasoning in natural language question answering. *AI Communications*, 23:241–265, 2010.

[16] John H. Gennari, Mark A. Musen, Ray W. Fergerson, William E. Grosso, Monica Crubzy, Henrik Eriksson, Natalya F. Noy, and Samson W. Tu. The evolution of protégé: An environment for knowledge-based systems development. *International Journal of Human-Computer Studies*, 58:89–123, 2002.

[17] Sanda Harabagiu, Dan Moldovan, Marius Paşca, Rada Mihalcea, Mihai Surdeanu, Zvan Bunescu, Roxana Girju, Vasile Rus, and Paul Morarescu. Falcon: Boosting knowledge for answer engines. In *Proceedings of the Ninth Text REtrieval Conference (TREC-9)*, pages 479–488, 2000.

[18] L. Hirschman and R. Gaizauskas. Natural language question answering: the view from here. *Natural Language Engineering*, 7(4):275–300, 2001.

[19] Valentin Jijkoun and Maarten 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*, CIKM '05, pages 76–83, 2005.

[20] Boris Katz. Annotating the world wide web using natural language. In *Proceedings of the 5th RIAO Conference on Computer Assisted Information Searching on the Internet - RIAO 1997*, pages 136–159, 1997.

[21] Oleksandr Kolomiyets and Marie-Francine Moens. A survey on question answering technology from an information retrieval perspective. *Information Sciences*, 181(24):5412–5434, 2011.

[22] Xin Li and Dan Roth. Learning question classifiers. In *Proceedings of the 19th international conference on Computational linguistics - Volume 1*, COLING '02, pages 1–7, 2002.

[23] Zhao Liu, Xipeng Qiu, Ling Cao, and Xuanjing Huang. Discovering logical knowledge for deep question answering. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, CIKM '12, pages 1920–1924, 2012.

[24] V Lopez, V Uren, M Sabou, and E Motta. Is question answering fit for the semantic web?: a survey. *Semantic Web*, 2:125–155, 2011.

[25] Vanessa Lopez, Victoria Uren, Enrico Motta, and Michele 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.

[26] Vanessa Lopez, Miriam Fernández, Enrico Motta, and Nico Stieler. Poweraqua: Supporting users in querying and exploring the semantic web. *Semantic Web*, 3(3): 249–265, 2012.

[27] Paul Martin, Douglas E. Appelt, Barbara J. Grosz, and Fernando Pereira. Team: an experimental transportable natural-language interface. In *Proceedings of 1986 ACM Fall joint computer conference*, ACM '86, pages 260–267, 1986.

[28] Deborah L. McGuinness. Question answering on the semantic web. *IEEE Intelligent Systems*, 19(1):82–85, 2004.

[29] Ana Cristina Mendes and Luísa Coheur. When the answer comes into question in question-answering: survey and open issues. *Natural Language Engineering*, pages 1–32, 2012.

[30] Michael Minock. C-phrase: A system for building robust natural language interfaces to databases. *Data & Knowledge Engineering*, 69(3):290–302, 2010.

[31] 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 (TREC-2002)*, 2002.

[32] Dan Moldovan, Sanda Harabagiu, Marius Pasca, Rada Mihalcea, Roxana Girju, Richard Goodrum, and Vasile Rus. Lasso: A tool for surfing the answer net. In *Proceedings of the 8th Text REtrieval Conference (TREC-8)*, 1999.

[33] Anh Kim Nguyen and Huong Thanh Le. Natural language interface construction using semantic grammars. In *Proceedings of the 10th Pacific Rim International Conference on Artificial Intelligence: Trends in Artificial Intelligence*, PRICAI '08, pages 728–739, 2008.

[34] Dai Quoc Nguyen, Dat Quoc Nguyen, and Son Bao Pham. A Vietnamese Question Answering System. In *Proceedings of the 2009 International Conference on Knowledge and Systems Engineering*, KSE '09, pages 26–32, 2009.

[35] Dang Tuan Nguyen and Tri Phi-Minh Nguyen. A question answering model based evaluation for ovl (ontology for vietnamese language). *International Journal of Computer Theory and Engineering*, 3(3), 2011.

[36] Dat Quoc Nguyen, Dai Quoc Nguyen, and Son Bao 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] Dat Quoc Nguyen, Dai Quoc Nguyen, Son Bao Pham, and Dang Duc 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*, CICLing'11, pages 190–201, 2011.

[38] Dat Quoc Nguyen, Dai Quoc Nguyen, and Son Bao Pham. KbQAS: A Knowledge-based QA System. In *Proceedings of the ISWC 2013 Posters & Demonstrations Track*, pages 109–112, 2013.

[39] Dat Tien Nguyen, Tam Duc Hoang, and Son Bao Pham. A vietnamese natural language interface to database. In *Proceedings of the 2012 IEEE Sixth International Conference on Semantic Computing*, pages 130–133, 2012.

[40] Anselmo Peñas, Bernardo Magnini, Pamela Forner, Richard Sutcliffe, Álvaro Rodrigo, and Danilo Giampiccolo. Question answering at the cross-language evaluation forum 2003-2010. *Language Resources and Evaluation*, 46(2):177–217, 2012.

[41] Dang Duc Pham, Giang Binh Tran, and Son Bao 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*, KSE'09, pages 154–161, 2009.

[42] Son Bao Pham and Achim Hoffmann. Efficient knowledge acquisition for extracting temporal relations. In *Proceedings of the 2006 conference on ECAI 2006: 17th European Conference on Artificial Intelligence, August 29 – September 1, 2006, Riva del Garda, Italy*, pages 521–525, 2006.

[43] T.T. Phan and T.C. 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 (ACIIDS2010)*, pages 29–40, 2010.

[44] Ana-Maria Popescu, Oren Etzioni, and Henry Kautz. Towards a theory of natural language interfaces to databases. In *Proceedings of the 8th international conference on Intelligent user interfaces*, IUI '03, pages 149–157, 2003.

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

[46] Fabio Rinaldi, James Dowdall, Kaarel Kaljurand, Michael Hess, and Diego Mollá. Exploiting paraphrases in a Question Answering system. In *Proceedings of the second international workshop on Paraphrasing - Volume 16*, PARAPHRASE '03, pages 25–32, 2003.

[47] Sanjay Silakari, Mahesh Motwani, and Neelu Nihalani. Natural language interface for database: A brief review. *IJCSI International Journal of Computer Science Issues*, 8:600–608, 2011.

[48] Eriks Sneiders. Automated question answering using question templates that cover the conceptual model of the database. In *Proceedings of the 6th International Conference on Applications of Natural Language to Information Systems-Revised Papers*, NLDB'02, pages 235–239, 2002.

[49] Niculae Stratica, Leila Kosseim, and Bipin 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] Marjorie Templeton and John Burger. Problems in natural-language interface to dbms with examples from eufid. In *Proceedings of the first conference on Applied natural language processing*, ANLP'83, pages 3–16, 1983.

[51] Mai-Vu Tran, Duc-Trong Le, Xuan-Tu Tran, and Tien-Tung 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.

[52] Christina Unger and Philipp Cimiano. Pythia: compositional meaning construction for ontology-based question answering on the semantic web. In *Proceedings of the 16th international conference on Natural language processing and information systems*, NLDB'11, pages 153–160, 2011.

[53] Christina Unger, Lorenz Bühmann, Jens Lehmann, Axel-Cyrille Ngonga Ngomo, Daniel Gerber, and Philipp Cimiano. Template-based question answering over rdf data. In *Proceedings of the 21st international conference on World Wide Web*, WWW '12, pages 639–648, 2012.

[54] Benjamin Van Durme, Yifen Huang, Anna Kupść, and Eric 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.

[55] M. Vargas-Vera and E. Motta. An ontology-driven similarity algorithm. Technical report, Knowledge Media Institute, The Open University, 2004.

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

[57] Ellen M. Voorhees. Overview of the trec 2001 question answering track. In *Proceedings of the Tenth Text REtrieval Conference (TREC-10)*, 2001.

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

[59] Ellen M. Voorhees. Overview of the trec 2002 question answering track. In *Proceedings of the 11th Text REtrieval Conference (TREC 2002)*, pages 115–123, 2002.

[60] David L. Waltz. An english language question answering system for a large relational database. *Commun. ACM*, 21(7):526–539, 1978.

[61] Chong Wang, Miao Xiong, Qi Zhou, and Yong 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] Bonnie Webber and Nick Webb. Question Answering. In *The Handbook of Computational Linguistics and Natural Language Processing*, pages 630–654. 2010.

[63] W. A. Woods, Ron Kaplan, and Nash B. Webber. The LUNAR sciences natural language information system: Final report. Technical Report BBN Report No. 2378, Bolt Beranek and Newman, 1972.

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

[65] Shiqi Zhao, Ming Zhou, and Ting Liu. Learning Question Paraphrases for QA from Encarta Logs. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, IJCAI'07, pages 1795–1801, 2007.

[66] Zhiping Zheng. Answerbus question answering system. In *Proceedings of the Second International Conference on Human Language Technology Research*, HLT '02, pages 399–404, 2002.