Systematic Knowledge Acquisition for Question Analysis

Dat Quoc Nguyen[†,♯] and **Dai Quoc Nguyen**[†,♯] and **Son Bao Pham**[†,‡]
[†] Faculty of Information Technology
University of Engineering and Technology
Vietnam National University, Hanoi
{datnq, dainq, sonpb}@vnu.edu.vn
[‡] Information Technology Institute
Vietnam National University, Hanoi

Abstract

For the task of turning a natural language question into an explicit intermediate representation of the complexity in question answering systems, all published works so far use rule-based approach to the best of our knowledge. We believe it is because of the complexity of the representation and the variety of question types and also there are no publicly available corpus of a decent size. In these rule-based approaches, the process of creating rules is not discussed. It is clear that manually creating the rules in an ad-hoc manner is very expensive and error-prone. In this paper, we focus on the process of creating those rules manually, in a way that consistency between rules is maintained and the effort to create a new rule is independent of the size of the current rule set. Experimental results are promising where our system achieves better performance and requires much less time and cognitive load compared to previous work.

1 Introduction

The goal of question answering systems is to give answers to the user's questions instead of ranked lists of related documents as used by most current search engines (Hirschman and Gaizauskas, 2001). Natural language question analysis component is the first component in any question answering systems. This component creates an intermediate representation of the input question, which is expressed in natural language, to be utilized in the rest of the system.

In this paper, we introduce a language independent approach to systematically build a knowledge base for analyzing natural language questions. Natural language questions will be transformed into intermediate representation elements which include construction type of question, class of question, keywords in question and semantic constraints between them.

Some question answering systems such as Aqualog (Lopez et al., 2007) and Vietnamese question answering system (VnQAS) (Nguyen et al., 2009) manually defined a list of sequence pattern structures to analyze questions. As rules are created in an ad-hoc manner, these systems share a common difficulty in managing interaction between rules and keeping consistency. In our approach, we present an approach utilizing Ripple Down Rules (Compton and Jansen, 1990) (Richards, 2009) knowledge acquisition methodology to acquire rules in a systematic manner which avoids unintended interaction between rules.

In section 2, we provide some related works and describe our overall system architecture in section 3. We present our knowledge acquisition approach for question analysis in section 4. We describe our experiments in section 5. Discussion and conclusion will be presented in section 6.

2 Related works

2.1 Question analysis in question answering systems

Early NLIDB systems used pattern-matching technique to process user's question and generate corresponding answer (Androutsopoulos, 1995). A common technique for parsing input questions in NLIDB approaches is syntax analysis where a natural language question is directly mapped to a database query (such as SQL) through grammar rules. Nguyen and Le (Nguyen and Le, 2008) introduced a NLIDB question answering 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 user's question into syntax tree via CYK algorithm.

Recently, some question answering systems that used semantic annotations generated high results in natural language question analysis. A well known annotation based framework is GATE (Cunningham et al., 2002) which have been used in many question answering systems especially for the natural language question analysis module such as Aqualog (Lopez et al., 2007), QuestIO (Damljanovic et al., 2008), VnQAS (Nguyen et al., 2009).

Aqualog and VnQAS are ontology-based question answering systems for English and Vietnamese respec-

[♯] Both authors contributed equally to this work.

tively. Both systems take a natural language question and an ontology as its input, and return answers for users based on the semantic analysis of the question and the corresponding elements in the ontology. General architecture of these systems can be described as a waterfall model where a natural language question is mapped to an intermediate representation. The subsequent modules of the system process the intermediate representation to provide queries with respect to the input ontology. These systems perform semantic and syntactic analysis of the input question through the use of processing resources wrapped as GATE plug-ins such as word segmentation, sentence segment and part-of-speech tagging.

2.2 Single Classification Ripple Down Rules

In this section we present the basic idea of Ripple-Down Rules (RDR) (Compton and Jansen, 1990) which inspired our approach. RDR allows one to add rules to a knowledge base incrementally without the need of a knowledge engineer. A new rule is only created when the KB performs unsatisfactorily on a given case. The rule represents an explanation for why the conclusion should be different from the KB's conclusion on the case at hand.

A Single Classification Ripple Down Rules (SCRDR) tree is a binary tree with two distinct types of edges. These edges are typically called *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 (a sentence to be classified in our case for example) to the root of the tree. At any node in the tree, if the condition of a node N's rule is satisfied by the case, the case is passed on to the exception child of N using the *except* link if it exists. Otherwise, the case is passed on to the N's *if-not* child. The conclusion given by this process is the conclusion from the last node in the RDR tree which *fired* (satisfied by the case). 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.

A new node 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* rule. Otherwise, it is attached with the *if-not* link.

RDR based approaches have been used to tackle NLP tasks such as POS tagging (Nguyen et al., 2011), text classification and information extraction (Pham and Hoffmann, 2006).

3 Our Question Answering System Architecture

The architecture of our question answering system is shown in Figure 1. It includes two components: the Natural language question analysis engine and the Answer retrieval.

The question analysis component consists of three modules: preprocessing, syntactic analysis and semantic analysis. It takes the user question as an input and returns a query-tuple representing the question in a compact form. The role of this intermediate representation is to provide structured information of the input question for later processing such as retrieving answers. Our contribution focuses on the semantic analysis module by proposing a rule language and a systematic processing to create rules in a way that interaction between rules are controlled and consistency are maintained.

Similar to VnQAS (Nguyen et al., 2009), 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.

To set the context for the discussion on the systematic knowledge acquisition process in the semantic analysis module, we will describe our question analysis component in details.

We wrapped existing linguistic processing modules for Vietnamese such as Word Segmentation, Part-of-speech tagger (Pham et al., 2009) as GATE plug-ins. 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 feature *category* storing its part-of-speech tag. This information can then be reused for further processing in subsequent modules. New modules are specifically designed to handle Vietnamese questions using patterns over existing linguistic annotations. This is achieved using GATE JAPE (Java Annotation Pattern Engine) transducers, a set of JAPE grammars. A JAPE grammar allows one to specify regular expression pattern based on semantic annotations.

3.1 Preprocessing module

The preprocessing module generates *TokenVn* annotations representing a Vietnamese word with features such as part-of-speech. 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 question phrases, which are indicative of the question categories such as *"phải không"*, tagged as multiple *TokenVn* annotations. In this module we identify those phrases and mark them as single annotations with corresponding feature *"question-word"* and its semantic categories such as $HowWhy_{cause \mid method}$, $YesNo_{true\ or\ false}$, $What_{something}$, $When_{time \mid date}$, $Where_{location}$, $Many_{number}$, Who_{person}. In fact, this information will be used in creating rules in the syntactic analysis module at a later stage.

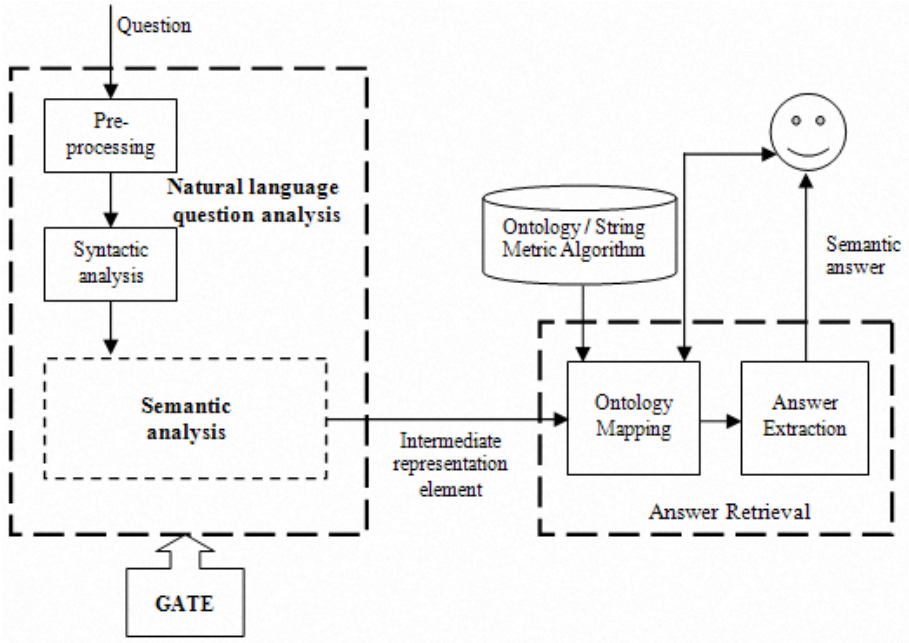

Figure 1: Architecture of our question answering system.

In addition, we marked phrases that refer to comparing-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.

3.2 Syntactic analysis

This module is responsible for identifying noun phrases and the relations between noun phrases. The different modules communicate through the annotations, for example, this module uses the *TokenVn* annotations, which is the result of the preprocessing module.

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*. We use *JAPE* grammars to specify patterns over annotations. When a noun phrase is matched, an annotation *NounPhrase* is created to mark up the noun phrase. In addition, its *type* feature is used to identify the concept and entity that is 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.

In addition, question-phrases are detected by using noun phrases and question-words identified by the preprocessing module. *QUTerm* or *QU-E-L-MC* annotations are generated to cover question-phrases with 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.

For example, with the following question:

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

"list all students whose hometown is Hanoi?"

The phrase *"có quê quán ở*$_{have\ hometown\ of}$*"* 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 Semantic analysis module

The semantic analysis module identifies the query-tuples to generate the intermediate representation of the input question using the annotations generated by the previous modules. We will present a systematic knowledge acquisition approach by building a SCRDR KB of rules in the next section.

4 Ripple Down Rules for Question Analysis

Unlike existing approaches for question analysis for English (Lopez et al., 2007) and Vietnamese (Nguyen et al., 2009) where manual rules are created in an ad-hoc manner, we will describe a language independent approach to analyze natural language questions by applying Ripple Down Rules methodology to acquire rules incrementally. Rules are structured in an exception-structure and new rules are only added to correct errors of existing rules.

A SCRDR knowledge base is built to identify the question structure and to produce the query-tuples as the intermediate representation. Figure 2 shows the GUI of our natural language question analyzer. We will first describe the intermediate representation used in our approach, and then propose a rule language for extracting this intermediate representation for a given input question.

4.1 Intermediate Representation of an input question

Aqualog (Lopez et al., 2007) performs semantic and syntactic analysis of the input English question through the use of processing resources provided by GATE (Cunningham et al., 2002). When a question is asked, the task of the question analysis 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. Following VnQAS (Nguyen et al., 2009), the intermediate representation used in our approach is more complex aiming to cover a wider variety of question types. It consists of a *question-structure* and one or more *query-tuple* in the following format:

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

where $Term_1$ represents a concept (object class), $Term_2$ and $Term_3$, if exist, represent entities (objects), *Relation* (property) is a semantic constraint between terms in the question. This representation is meant to capture the semantic of the question.

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

"danh sách tất cả các sinh viên của khoa công nghệ thông tin mà có quê quán ở Hà Nội?"

"list all students in the Faculty of Information Technology whose hometown is Hanoi?"

has question structure of type *And* with two query-tuples where **?** represents a missing element: *(UnknRel , List , sinh viên*$_{student}$ *, ? , khoa công nghệ thông tin*$_{Faculty\ of\ Information\ Technology}$ *, ?)* and *(Normal , List , sinh viên*$_{student}$ *, có quê quán* $_{has\ hometown}$ *, Hà Nội*$_{Hanoi}$ *, ?)*.

This representation is chosen so that it can represent a richer set of question types. Therefore, some terms or relation in the tuple can be missing. Existing noun phrase annotations and relation annotations are potential candidates for terms and relations respectively. Following VnQAS (Nguyen et al., 2009), 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*.

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 (Cunningham et al., 2002). It can also post new annotations over matched phrases of the pattern's sub-components. The following example of a pattern shows the posting an annotation over the matched phrase:

(({TokenVn.string == "liệt kê$_{list}$"} |
{TokenVn.string == "chỉ ra$_{show}$"})
{NounPhrase.type == Concept}) : QU_LIST

This 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 NounPhrase which must have feature type equal to Concept. When applying this pattern on a text fragment, QU_LIST annotations would be posted over phrases matching this pattern. 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 rule's conclusion contains the question structure and the tuples corresponding to the intermediate representation where each element in the tuple is specified by a newly posted annotations from matching the rule's condition in the following order:

(question-structure, question-class, $Term_1$*, 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 section.

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 such as TokenVn, NounPhrase, Relation, annotations capturing question-phrases like QUTerm, QU-E-L-MC (Entity, List, ManyClass)... and their features.

4.3 Knowledge Acquisition Process

The following examples show how the knowledge base building process works. When we encountered the question:

"trường đại học Công Nghệ có bao nhiêu sinh viên?" (*"how many students are there in the College of Technology?"*)

[NounPhrase trường đại học Công Nghệ$_{the\ College\ of\ Technology}$ NounPhrase][Has có$_{has}$ Has] [QU-E-L-MC bao nhiêu sinh viên$_{how\ many\ students}$ QU-E-L-MC]

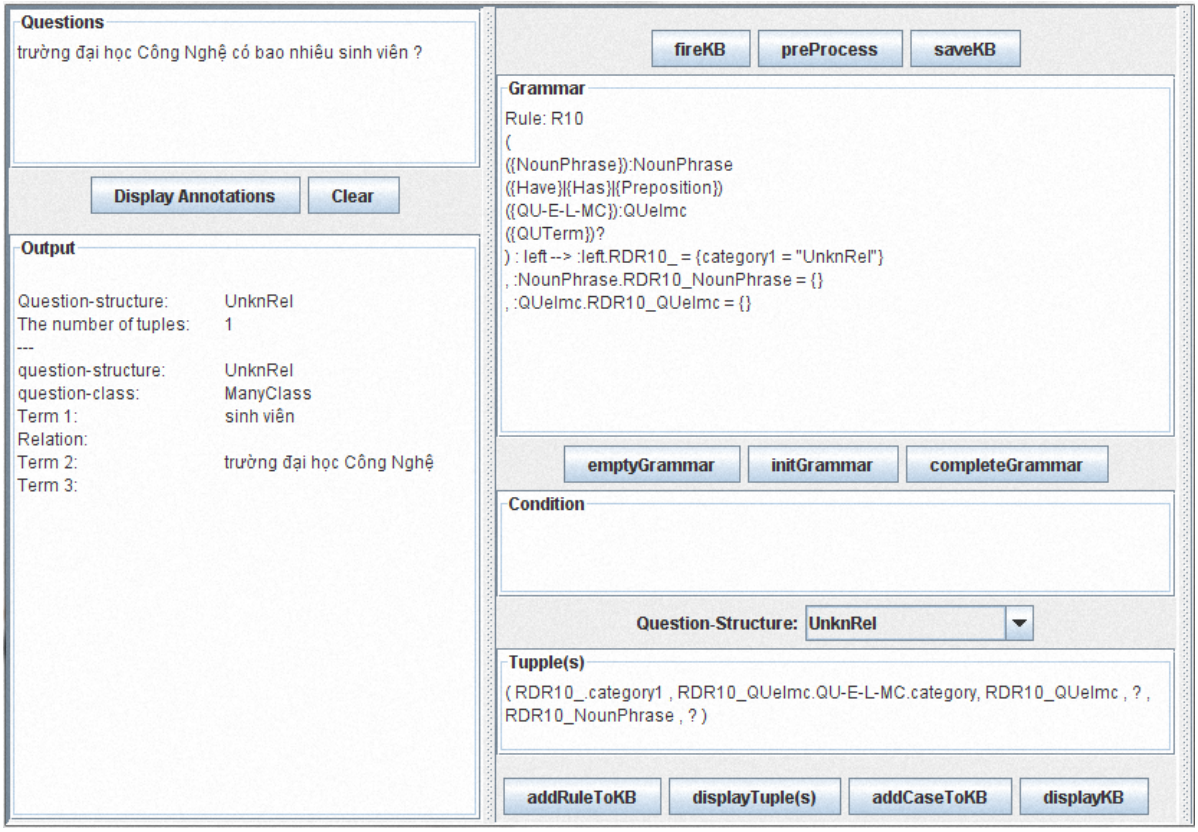

Figure 2: Question Analysis module to create the intermediate representation of question *"trường đại học Công Nghệ có bao nhiêu sinh viên?"*(*"how many students are there in the College of Technology?"*).

Supposed we start with an empty knowledge base, the fired rule is default rule that gives empty conclusion. This can be corrected by adding the following rule to the knowledge base:

Rule: R10
(
({NounPhrase}):NounPhrase
({Have}|{Has}|{Preposition})
({QU-E-L-MC}):QUelmc
({QUTerm})?
) : left --→ :left.RDR10_ = {category1 = "UnknRel"}
, :NounPhrase.RDR10_NounPhrase = {}
, :QUelmc.RDR10_QUelmc = {}

Conclusion: question-structure of *UnknRel* and tuple *(RDR10_.category1 , RDR10_QUelmc.QU-E-L-MC.category, RDR10_QUelmc , ? , RDR10_NounPhrase , ?)*.

If the condition of rule **R10** matches the whole input question, a new annotation RDR10_ will be created covering the whole input question and new annotations RDR10_NounPhrase and RDR10_QUelmc will be created to cover sub-phrases of the input question.

If rule **R10** is fired, the matched input question is deemed to have a query-tuple with question-structure taking the value of *category1* feature of RDR10_ annotation, question-class taking the value of *category* feature of QU-E-L-MC annotation co-covering the same span as RDR10_QUelmc annotation, $Term_1$ is the string covered by RDR10_QUelmc, $Term_2$ is the string covered by RDR10_NounPhrase while $Term_3$ and *Relation* are unknown.

When we encounter the question:

"trường đại học Công Nghệ có bao nhiêu sinh viên là Nguyễn Quốc Đạt?" (*"How many students named Nguyen Quoc Dat are there in the College of Technology?"*)

[RDR10_ *trường đại học Công Nghệ có bao nhiêu sinh viên* RDR10_] [Are là$_{Are}$ Are] [NounPhrase Nguyễn Quốc Đạt$_{Nguyen\ Quoc\ Dat}$ NounPhrase]

Rule **R10** is the fired rule but gives the wrong conclusion of question-structure of *UnknRel* and tuple *(UnknRel , ManyClass , sinh viên$_{student}$, ? , trường đại học Công Nghệ$_{the\ College\ of\ Technology}$, ?)*. The following exception rule was added to knowledge base to correct that:

Rule: R38
(
{RDR10_} ({Are}|{Is})
({NounPhrase}):NounPhrase
):left --→ :left.RDR38_ = {category1 = "ThreeTerm"}
, :NounPhrase.RDR38_NounPhrase = {}

Conclusion: question-structure of *ThreeTerm* and tuple *(RDR38_.category1 , RDR10_QUelmc.QU-E-L-MC.category , RDR10_QUelmc , ? , RDR10_NounPhrase , RDR38_NounPhrase)*.

Using rule **R38**, the output of the input question is question-structure of *ThreeTerm* and tuple *(ThreeTerm , ManyClass , sinh viên$_{student}$, ? , trường đại học Công Nghệ$_{the\ College\ of\ Technology}$, Nguyễn Quốc Đạt$_{Nguyen\ Quoc\ Dat}$)*

With the question *"quê quán của những sinh viên nào là Hà Nội?"* (*"which students have hometown of Hanoi?"*)

[RDR10_ [RDR10_NounPhrase *quê quán$_{hometown}$* RDR10_NounPhrase] [Preposition *của$_{of}$* Preposition] [RDR10_QUelmc *những sinh viên nào$_{which\ students}$* RDR10_QUelmc] RDR10_][Are *là$_{are}$* Are] [RDR38_NounPhrase *Hà Nội$_{Hanoi}$* RDR38_NounPhrase]

it will be satisfied by rule **R38**. But rule **R38** gives the wrong conclusion of question-structure of *ThreeTerm* and tuple *(ThreeTerm , Entity , sinh viên$_{student}$, ? , quê quán$_{hometown}$, Hà Nội$_{Hanoi}$)* because *quê quán$_{hometown}$* is a relation for linking *sinh viên$_{student}$* and *Hà Nội$_{Hanoi}$*. We can add a following exception rule **R76** to correct the conclusion by using constrains via rule condition:

Rule: R76

({RDR38_}):left

--→ :left.RDR76_ = {category1 = "Normal"}

Condition: RDR10_NounPhrase.hasAnno == NounPhrase.type == Concept

Conclusion: question-structure of *Normal* and tuple *(RDR76_.category1 , RDR10_QUelmc.QU-E-L-MC.category , RDR10_QUelmc , RDR10_NounPhrase , RDR38_NounPhrase , ?)*

The condition of rule **R76** matches a RDR10_NounPhrase annotation that has a NounPhrase annotation covering their substring with *Concept* as its *type* feature. The extra annotation constrain *hasAnno* requires that the text covered by the annotation must contain the specified annotation. With the rule **R76**, we have the correct output containing the question-structure of *Normal* and tuple *(Normal , Entity , sinh viên$_{student}$, quê quán$_{hometown}$, Hà Nội$_{Hanoi}$, ?)*.

5 Experiments

We experiment our system for both Vietnamese and English using the same intermediate representation.

5.1 Question Analysis for Vietnamese

For this experiment, we build a knowledge base of 92 rules from a corpus containing 400 questions and evaluate its quality on an unseen corpus of 102 questions in the same domain of college (university). The corpus of 400 questions were generated based on a seed corpus of 115 questions. Table 1 shows 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 that is not an exception rule, is the default rule in layer 0. This indicates that the exception structure is indeed present and even extends to level 4.

Table 1: 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 implemented the question analysis component of VnQAS (Nguyen et al., 2009) on the same corpus as in building our knowledge base. Table 2 shows the number of correctly analyzed questions of our system and system of (Nguyen et al., 2009) respectively where our system performs slightly better.

Table 2: Number of correctly analyzed questions

Type	Number of questions	Percent
Our system	88	86.3%
Question analysis component of (Nguyen et al., 2009)	83	81.4%

Our method took one expert about 13 hours to build a KB based on the training corpus. 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 of VnQAS (Nguyen et al., 2009) took about 75 hours for creating rules in an ad-hoc manner. Anecdotal account indicates that the cognitive load in creating rules in our approach is much less compared to that in VnQAS (Nguyen et al., 2009) as in our case, we do not have to consider other rules when crafting a new rule.

Table 3 shows the source of error for the 14 questions that our system incorrectly extract. It clearly shows 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 3: Error results

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

5.2 Question Analysis for English

For the experiment in English, we take 170 English question examples of AquaLog's corpus. Using our ap-

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

proach, 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. The table 4 shows the numbers of rules in English knowledge base layers.

Table 4: 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

As the intermediate representation of our system is different to Aqualog and there is no common test set available, it is impossible to directly compare our approach with Aqualog on the English domain. However, this experiment is indicative of the ability in using our system to quickly build a new knowledge base for a new domain and a new language.

6 Conclusion

We believe 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 using our system. Incrementally, an annotated corpus and a rule-based system can be obtained simultaneously.

The structured data used in the 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.

In this paper, we introduced a language independent approach for systematically acquiring rules for converting a natural language question into an intermediate representation in a question answering system. Experimental results of our system on a wide range of questions are promising with accuracy of 86.3% for the Vietnamese corpus. Notably, the time it takes to get the system up to this performance is much less compared to previous works.

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.

Acknowledgements

This work is partially supported by the Research Grant from Vietnam National University, Hanoi No. QG.10.23.

The authors would like to acknowledge Vietnam National Foundation for Science and Technology Development (NAFOSTED) for their financial support to present the work at the conference.

References

L. Androutsopoulos. 1995. Natural language interfaces to databases - an introduction. *Nat. Lang. Eng.*, 1:29–81.

P. Compton and R. Jansen. 1990. A philosophical basis for knowledge acquisition. *Knowl. Acquis.*, 2(3):241–257.

Hammish Cunningham, Diana Maynard, Kalina Bontcheva, and Valentin Tablan. 2002. GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications. In *Proc. of ACL*, pages 168–175.

Danica Damljanovic, Valentin Tablan, and Kalina Bontcheva. 2008. A text-based query interface to owl ontologies. In *Proc. of LREC*, pages 205–212.

L. Hirschman and R. Gaizauskas. 2001. Natural language question answering: the view from here. *Nat. Lang. Eng.*, 7(4):275–300.

Vanessa Lopez, Victoria Uren, Enrico Motta, and Michele Pasin. 2007. Aqualog: An ontology-driven question answering system for organizational semantic intranets. *Web Semant.*, 5(2):72–105.

Anh Kim Nguyen and Huong Thanh Le. 2008. Natural language interface construction using semantic grammars. In *Proc. of PRICAI*, pages 728–739.

Dai Quoc Nguyen, Dat Quoc Nguyen, and Son Bao Pham. 2009. A vietnamese question answering system. In *Proc. of KSE*, pages 26–32.

Dat Quoc Nguyen, Dai Quoc Nguyen, Son Bao Pham, and Dang Duc Pham. 2011. Ripple down rules for part-of-speech tagging. In *Proc. of CICLing*, pages 190–201.

Son Bao Pham and Achim Hoffmann. 2006. Efficient knowledge acquisition for extracting temporal relations. In *Proc. ECAI*, pages 521–525.

Dang Duc Pham, Giang Binh Tran, and Son Bao Pham. 2009. A hybrid approach to vietnamese word segmentation using part of speech tags. In *Proc. of KSE*, pages 154–161.

Debbie Richards. 2009. Two decades of ripple down rules research. *Knowl. Eng. Rev.*, 24(2):159–184.