An Experimental Study of Vietnamese Question Answering System

Vu Mai Tran, Vinh Duc Nguyen, Oanh Thi Tran
Information Systems Dept, College of Technology,
VNU Hanoi, Vietnam
Email: {vutranmai, vinhndk,oanhtt}@vnu.edu.vn

Uyen Thu Thi Pham , Thuy-Quang Ha1
Information Systems Dept, College of Technology,
VNU Hanoi, Vietnam
Email: {phamthuuyen, thuyhq}@vnu.edu.vn

Abstract—The development of World Wide Web calls for how to efficiently exploit the information. Mostly, current search engines return a set of related documents which contain keywords. However, users expect the exact and concrete answer for each question. Therefore, it is necessary to build an automatic question answering system (QA). In this paper, we focus on building a QA for Vietnamese. This task especially becomes more and more difficult because of the lack of available tools for processing Vietnamese text. Based on previous research for English, this paper proposed an implementation for Vietnamese question answering system by combining SnowBall system [1] and semantic relation extraction using search engines [4]. The experimental results on travelling domain proved that this method is feasible for Vietnamese question answering system. We achieved 89.7% precision and 91.4% ability to give the answers when testing on travelling domain.

Keywords: Vietnamese question answering, semantic relation extraction, snowball, search engine

I. INTRODUCTION

Nowadays, question answering system (QA) has attracting many researcher as well as companies all over the world. Many international conferences on Natural Language Processing such as ACL, COLING, TREC have separated tracks on researches about QA. In addition, there are commercial softwares relating to QA also developed like Yahoo Answers of Yahoo, Google Question of Google (this product was only developed in Russian), Live QnA of Microsoft,… and especially there are two softwares (this product was only developed in Russian), Live QnA of Microsoft,… and especially there are two softwares which are Answers.com of Answer Corp achieved 9.5 million USD and Ask of InterActive Corp gained 227 million USD annually.

There are many methods proposed for building QA systems. In 1997, START online system used a database to analyze questions and give answers [5]. In 2004, Kim et al. presented OntotrileQA using semantic relation extraction technique built by hand for entities of ontology [3]. Most state-of-the-art QA systems utilize semantic relation extraction technique. For example, in 2002 Ravichandran and Hovy proposed extracting semantic relation using search engines [4]. In 2009, Fahmi increased the coverage of relations by using semi-supervised learning to automatically generate relations from a large set of data [2].

In Vietnam, there is no question answering system currently. This is partly due to the lack of available and good enough tools for processing Vietnamese like Named Entity Recognition, Syntax Parsing and so on. Based on previous researches on Question Answering for English, we investigated and proposed a model for Vietnamese QA system. From many methods, we realized that semantic relation extraction method is the most suitable for Vietnamese. Therefore, in this paper we use semantic relation extraction method by combining pattern extraction in SnowBall system [1] and search engine method [4].

The experimental results on travelling domain proved that this method is feasible for Vietnamese QA. We achieved 89.7% precision and 91.4% ability to give the answer when testing on travelling domain.

The remaining of this paper is organized as following. In the next section, we present a model for Vietnamese question answering system. And in section 3, we present an experiment on travelling domain. Finally, in section 4, we give conclusion and offer the future work.

II. A PROPOSED MODEL FOR VIETNAMESE QUESTION ANSWERING SYSTEM

Figure 1 present the architecture for question answering system. There are two main phases Extracting Patterns and Tuples and Processing Question and Extracting Answer.

A. Phase 1. Extracting Patterns and Tuples

Input:
- Pre-defined relations
- Several seeds of each relation to support for bootstrapping learning

Output:
- Refined patterns and tuples which are used in phase 2

Step 1: Crawling snippets
For each seed <A, B> of relation R, we create queries and feed to a search engine. We obtain text snippets containing both A and B. It is more useful to use wildcard characters in finding such snippets by using search engines that support wildcard search. More
specifically, given a seed \(<A, B\>)\), the system produces several queries in forms of “\(A \ast B\)”\), “\(A \ast * B\)”\), “\(B \ast * A\)”\). Such snippets are very effective for generating patterns due to the fact that the occurrences of \(A\) and \(B\) are close to each other.

Based on these snippets, the system finds occurrences - sentences that contain both \(A\) and \(B\). These occurrences are used to generate patterns in the next step.

**Step 2: Generating patterns**

A critical step in this phase is generating refined patterns in order to find new tuples. Each pattern composes of five parts \(<\text{left}, \text{tag}_1, \text{middle}, \text{tag}_2, \text{right}>\) where \(\text{tag}_1, \text{tag}_2\) are named-entity tags and \(\text{left}, \text{middle}, \text{right}\) are weighted term vectors. In our system, each term is a word consisting of one or more morphemes which are separated by white space. Term’s weight is frequency of term in the corresponding context.

Firstly, the system creates rough patterns from occurrences. The system should identify two entities in each occurrence and analyze the contexts (strings) to create three vectors called \(\text{left}, \text{middle}, \text{right}\) respectively. It is noted that the middle string is more important than the left and the right one. Therefore, when creating these vectors, we use all terms in middle string but only a restricted number of terms in the left and the right. In our implementation, this restricted number is set to 5.

To generate refined patterns, the system clusters similar rough patterns. Each rough pattern is clustered to a cluster of patterns using simple algorithm called single-pass clustering algorithm [7]. The similarity between two patterns is defined as below Match function:

\[
 Match(p_1, p_2) = p_1.\text{left} \ast p_2.\text{left} + p_1.\text{middle} \ast p_2.\text{middle} + p_1.\text{right} \ast p_2.\text{right}
\]

if \(p_1.\text{Tag}_1 = p_2.\text{Tag}_1\) and \(p_1.\text{tag}_2 = p_2.\text{tag}_2\).

The patterns which are the centroids of clusters are called refined patterns. Such patterns are then used to generate new tuples.

**Step 3: Generating new tuples**

The system creates queries from these refined patterns and uses Google search engine to get snippets that contain new tuples. These snippets are tagged by a named-entity tagger. After that, sentences which contain both entities of the relation are determined. Next, the system generates a 5-part tuple \(t = \langle \text{left}, t_1, \text{middle}, t_2, \text{right}\rangle\) where \(t_1, t_2\) are name entities. A rough tuple \((t_1, t_2)\) is generated if there is a refined pattern \(p\) that satisfies \(Match(p, t) \geq p_i\) where \(p_i\) is the similarity threshold used in clustering.

In this step, evaluating patterns and tuples is the same as SnowBall system’s [1].

**B. Phase 2. Processing Question and Extracting Answer**

The system is supported to answer simple questions about relations between entities. The question processing phase has to identify related entity occurring in the question and relation. To identify entity in the question, the system uses a named-entity tagger. To identify relation that question asks about, the system computes the similarity between question and patterns generated in previous phase and choose the best match pattern. Relation \(R\) that the pattern belongs to is the relation that question asks about.

Based on tuples generated in previous phase, the system provides answer by querying database. The record returned contains relation \(R\) and an entity in the question.

**C. Recognizing Vietnamese named entities**

In both two phases, the system requires a named-entity tagger. However, there is not any good public NER system for recognizing Vietnamese entities currently. In more detail, the system needs to recognize specific named entities such as FESTIVAL, MOUNTAIN, RIVER, BEACH, and so on. These entities, nevertheless, haven’t supported by any general NER. Therefore, we have to find another effective method to automatically determine such entities. In this system, we use iSEAL (Iterative Set Expander for Any Language) [6] to expand entities. iSEAL expands entities automatically by utilizing available resources from the Web in a language-independent fashion. The inputs for iSEAL are seeds belong to a user-desired class (TAG). iSEAL uses these inputs for bootstrapping learning and then outputs other instances of that class. Then, iSEAL system continue using user-provided seeds and self-generated seeds for the next iterative. Finally, iSEAL uses these entities for entity tagging.

**III. A CASE STUDY IN QUESTION ANSWERING FOR TRAVELING DOMAIN**

In this paper, we implement this model on traveling domain. The system takes simple questions about traveling such as “Hội Lim được tổ chức ở đâu?” (“Where is Lim festival held?”) and provides a concise answer: “tỉnh Bạc Ninh” (“Bac Ninh province”).

**A. Investigation of traveling domain**

In our system, we determined ~ 85 relations in this domain. After that, we chose 10 popular relations for the experiment. Each relation presents a relationship between two entity classes, for example “FESTIVAL → được tổ chức ở – PLACE” (Festival-Place for short).

For each relation, we provide 5 to 10 seeds for bootstrapping learning in phase 1

**B. Implementing set expansion of named entities using Web**

For each entity tag, the system is fed with several seed entities. The system uses these entities for bootstrapping learning to expand this entity set.

<table>
<thead>
<tr>
<th>Initial tagged entities</th>
<th>Top 5 generated entities</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hội chùa Hương (Huong pagoda festival)</td>
<td>Hội chùa thay (Thay pagoda festival)</td>
</tr>
<tr>
<td>Hội đèn Hùng (Hung temple festival)</td>
<td>Hội đèn Thường (Thuong temple festival)</td>
</tr>
<tr>
<td>Hội Lim (Lim festival)</td>
<td>Hội chùa Keo (Keo pagoda festival)</td>
</tr>
<tr>
<td>Hội đèn Chùa Đồng Tứ</td>
<td>Hội chùa Hỏa (Hoa pagoda festival)</td>
</tr>
</tbody>
</table>
C. Implementing pattern and tuple extraction

- Generating Patterns:
  Setting up pattern similarity threshold: This parameter controls how flexible the patterns are. In our experiment, this threshold is set to 0.5.
  - Extracting Tuples:
    Setting up Tuple Confidence Threshold: This threshold determines the minimum confidence that a tuple will be selected into the seed set for the next iteration. In our experiment, this threshold is set to 0.6.

Table 2 shows results for 10 relations with rough patterns, refined patterns and generated tuples.

Table 2. The results of some relations are implements

<table>
<thead>
<tr>
<th>Relations</th>
<th>#Initia l seeds</th>
<th>#rough patterns</th>
<th>#refined patterns</th>
<th>#Obtaine d seeds</th>
</tr>
</thead>
<tbody>
<tr>
<td>Festival – Place</td>
<td>10</td>
<td>509</td>
<td>431</td>
<td>194</td>
</tr>
<tr>
<td>Beach – Place</td>
<td>8</td>
<td>3022</td>
<td>1720</td>
<td>203</td>
</tr>
<tr>
<td>Pagoda – Place</td>
<td>7</td>
<td>1034</td>
<td>756</td>
<td>462</td>
</tr>
<tr>
<td>River – Place</td>
<td>7</td>
<td>256</td>
<td>145</td>
<td>57</td>
</tr>
<tr>
<td>Café – Place</td>
<td>8</td>
<td>345</td>
<td>314</td>
<td>236</td>
</tr>
<tr>
<td>Restaurant – Place</td>
<td>8</td>
<td>389</td>
<td>354</td>
<td>563</td>
</tr>
<tr>
<td>Hotel – Place</td>
<td>8</td>
<td>245</td>
<td>213</td>
<td>346</td>
</tr>
<tr>
<td>Supermarket– Place</td>
<td>8</td>
<td>343</td>
<td>232</td>
<td>132</td>
</tr>
<tr>
<td>Park – Place</td>
<td>8</td>
<td>234</td>
<td>145</td>
<td>38</td>
</tr>
<tr>
<td>Market – Place</td>
<td>7</td>
<td>589</td>
<td>430</td>
<td>597</td>
</tr>
</tbody>
</table>

D. Question processing and answer extraction

Test data: we collect about 100 questions for 10 relations in travel domain.

The precision of our system depends on how well the system determines the relation of the question based on pattern sets. To provide a reliable answer, the system needs to find out a pattern that best matches with the question, and the match degree is greater than a threshold $\mu$. If there is no such pattern, the system will not provide answer. Table 3 shows ability to give answers with different thresholds $\mu$.

Table 3. The ability to give answers corresponding to different thresholds

<table>
<thead>
<tr>
<th>$\mu$</th>
<th>Precision</th>
<th>Ability to give answers</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.4</td>
<td>85.5%</td>
<td>95.3%</td>
</tr>
<tr>
<td>0.5</td>
<td>89.7%</td>
<td>91.4%</td>
</tr>
<tr>
<td>0.6</td>
<td>92.6%</td>
<td>80.3%</td>
</tr>
</tbody>
</table>

An effective question answering system must provide the most reliable answers (high precision) and answers a large number of questions. It is remembered that this precision is in inverse proportion to the ability to give answers. Through experiments, we choose $\mu = 0.5$ to balance the precision and the ability to give answers.

The following example gives a detailed description about steps of our system:

**Question: Nam Định có những bãi biển nào?**  
*(Which beach does Nam Định have?)*

**Step 1:** Recognizing named entity in the question.

Nam Định is recognized as an entity in the question.

**Step 2:** Determining the relation in the question. The system scans tuples that contain keyword “Nam Định”. From these tuples, the system finds out a set of candidate relations (see table 4).

**Step 3:** Extracting answer

In this example, the question vector is < cơ, bãi biển > and the best match pattern is

**Step 3:** Extracting answer

The system queries database by using information from two previous steps (relation R and entity E) to get tuple record(s) that is belong to R and have a field E. In For the above example, the system will output the answer: Quất Lâm, Hải Thịnh. Table 6 shows some question and our system’s answers accompanied with similarity thresholds.

System’s result is good for some simple questions which have semantic relation in travel domain. The precision is quite high. The system achieved 89.7% precision and 91.4% ability to give the answer.
### Table 6: Some questions and answers

<table>
<thead>
<tr>
<th>Questions</th>
<th>Answers</th>
<th>Best match patterns</th>
<th>Similarity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hà Tây cò lễ hội gì? (What festival is held in Hai Tay province?)</td>
<td>hội chùa Hương, hội chùa Thầy, hội đình cá lang me, hội đa ngư, hội lang có trai, ...</td>
<td>&lt;DIADIEM&gt; có lễ hội &lt;LEHOI&gt; &lt;PLACE&gt; has &lt;FESTIVAL&gt;</td>
<td>0.999</td>
</tr>
<tr>
<td>Lễ hội chùa Hương được tổ chức ở đâu? (Where is Huong festival held?)</td>
<td>Hà Tây</td>
<td>&lt;DIADIEM&gt; tổ chức lễ &lt;LEHOI&gt;</td>
<td>0.71</td>
</tr>
<tr>
<td>Bãi biển Cát bà thuộc thành phố nào? (What city does Cat Ba beach located in?)</td>
<td>Hải phòng</td>
<td>&lt;BAIBIEN&gt; thuộc &lt;DIADIEM&gt;</td>
<td>0.81</td>
</tr>
<tr>
<td>Hồ Ba bè ở đâu? (Where is Ba Be lake?)</td>
<td>Bắc Kạn</td>
<td>Hồ &lt;HO&gt; nằm ở &lt;DIADIEM&gt;</td>
<td>0.67</td>
</tr>
</tbody>
</table>

Throughout our experiments, it is shown that building model for relation extraction by combining Snowball and search engine methods is very promising for Vietnamese question answering system.

### IV. CONCLUSION AND FUTURE WORKS

In this paper, we presented an implementation for a Vietnamese question answering system. By investigating previous researches and the current status of current Vietnamese processing tools, we proposed a method using semantic relation extraction by combining SnowBall and search engine methods. The experiment results on travelling domain proved that this method is suitable for Vietnamese QA. The system achieved 89.7% precision and 91.4% ability to give the answer.

In the future, we will expand this QA system into other domains to build an open system supporting for end users.

**ACKNOWLEDGMENT**

This work was fully supported by the National project KC 01.02/06-10.

### REFERENCES


