Semantic parsing for Vietnamese Question Answering System

Vu Xuan Tung and Nguyen Le Minh

Japan Advanced Institute of Science and Technology
tungvx@jaist.ac.jp and nguyenml@jaist.ac.jp

1 Abstract

Based on the framework in [2], we developed a Vietnamese question answering system. The learning paradigm in [2] reduces the burden of providing supervision during semantic parsing. Whilst taking the advantages from this mechanism, we also create our own feature calculation which is suitable for Vietnamese. A method of dynamic learning for feature computation is also presented in this work.

2 Introduction

Semantic parsing is a very important process in Natural Language Processing. It converts a natural language sentence into a special meaning representation so that computer programs can read and process it. An effective way to do this task is using machine learning to build a model of relationship between natural language structure and formal representation structure. Unfortunately, most of the current approaches require very large amount of fully annotated data in order to obtain a good model. Annotated data means that for each input sentence of natural language, the corresponding representation is also provided. State-of-the-art methodologies often use statistical analysis to build the model and thus generally ill-perform with unseen data. Therefore, in order to get good results, they need to collect a large amount of annotated corpus. The annotation, however, is almost prepared manually and thus is difficult and time consuming. [7], [5], [8], [1], [9] and [6] are examples of works in this category.

Recently, James et al. in [2] implemented a new learning paradigm aimed at alleviating the supervision burden. The algorithm is able to predict complex structures which only rely on a binary feedback. Borrowing the idea from these authors, we developed a question answering system for Vietnamese with suitable feature computations.

Related Work

There has been not many works in Vietnamese semantic parsing. Nguyen et al. in [3] developed a question answering system for Vietnamese. The system enable users to query an ontological knowledge base using pattern matching. This is not really sematic parsing but semantic interpretation. The method here is simple and very difficult to scale to large database.

The rest of this paper is organized as follow. Section 3 describes how natural language sentences are represented formally in our semantic parser. Section 4 illustrates the way we maintain the parsing model. We present our experiment result and discuss it in Section 5. Finally, conclusion and future works are drawn in Section 6.

3 Meaning Representation

The representation of meaning (the result of semantic parsing) depends on each system and need not to be unified. Similar to "Target Meaning Representation" of [2], the representation in our system is the composition of pre-defined functions. Each function has only one argument. For example, $longest(river(stateId$ $(''colorado'')))$ is the formal representation of "con sông dài nhất chảy qua colorado là gì?" ("What is the longest river run through colarado ?"). There are two steps of building such a representation from a natural language sentence. The first one is finding the mapping between each token in natural language sentence to a function. Next, we need to compose the mapped functions into a complete logical form.

3.1 Predefined Functions

As mentioned, our meaning representation uses a set of pre-defined functions. Each function has only one parameter and returns a value of a type. In addition, a function should not accept any kind of argument, but only suitable typed one. For this reason, types of functions are supposed to be configured. For instance, *length* function receives an argument of type *River* and return an $Integer$ value. The list of functions and their information are defined in a text file. This way of configuration makes our system flexible because if we want to change the data, just changing the functions configuration file is enough.

3.2 Token - Function mapping

In Natural language processing, there often is a pre-process phase, in which a list of tokens is generated from the input sentence. This phase is also called tokenization. Token is an element that has some meanings. For example, "New Mexico" should be one token instead of being divided into two tokens. This is because "New Mexico" is the name of a city, we can not separate two words as different tokens. An other instance is "tiểu bang" in Vietnamese. This word means "state". If we break it into "tiểu" and "bang" as tokens, the original meaning is not preserved. Currently, there are some works which have dealed with tokenization for Vietnamese with high accuracies such as vnTokenizer in [4].

In our system, we assume that tokenization is provided by some other tools. Our accepted inputs are tokenized sentences. For example, "con sông, dài nhất, chảy, qua, colorado, là, gì, ?" is the accepted input, being translated into English as "What, is, the, longest, river, run, through, colarado, ?". Each token is supposed to be aligned with one predefined function. In the current considered example, "con sông", "dài nhất" and "colorado" are mapped to *river*, *longest* and *stateId* functions respectively.

3.3 Function - Function composition

When we have the alignment between tokens and functions, we need to make a final composition of the chosen functions. A function $f1$ could be composed with a function $f2$, i.e $f1(f2)$ is the result of composition, if returned type of $f2$ agrees with one of the argument types of $f1$. This is the reason why our system requires types of functions to be described in the configuration file. In previously mentioned example, *longest* is composed with *river*; *river* is composed with *stateId*. This leads to the final representation to be of the form $longest(river(stateId("colorado")))$.

4 Semantic parsing model

Our system is developed based on the framework in [2]. The process of building logical formulas from a natural language sentence is presented in [2]. As mentioned, each parser system has a model containing data to be used for translating a natural language sentence to formal representation. Similar to [2], model in our system consists of two parts: Word-Function feature calculator and Function-Function feature calculator.

4.1 Token-Function feature vector calculator

This component is in charge of computing feature vector given a token and a function. Each function is configured to have a set of surface forms, which are typically associated with the function. The feature vector is calculated by comparing the token with these surface forms. Thus, the larger the number of surface forms is, the more accurate the feature vector is. Similar to [2], we restrict the number of surface forms per function. In our current experiment, there is average 1.39 words per functions, in comparison with 1.42 of [2].

The feature vector is generated based on lexical matching between token and surface forms. Many cases in Vietnamese, two words with lexical similarity may have the same meaning. For example "Tiểu bang" and "bang" both have meaning as "state". Similary, we have "con sông" and "sông" (river), "ngọn núi" and "núi", etc.

4.2 Function - Function feature vector calculator

Suppose we need to compute the feature vector of f_1 and f_2 when we want to form $f_1(f_2(arg))$. arg here is the argument of $f2$. The most important note is that returned type of f_2 must agree with one of argument types of f_1. The second feature is the vector of distance between the tokens mapped to two functions in the sentence. If the distance is small, they will have more chances to be composed. This is due to the usual speaking style of Vietnamese. Let us consider the following example:
"con sông nào chảy qua tiểu bang missisipi?" ("what river runs through state mississippi?"). There are at least two candidates for the result of semantic parser. They are illustrated in figure 1 and figure 2. Between these two cases, our system tend to put more scores (higher value of features) on the first case. The intuition here is that, for example, "con sông" comes before "tiểu bang", the function $river$ is preferred to combine with $state$ to form $river(state(arg))$.

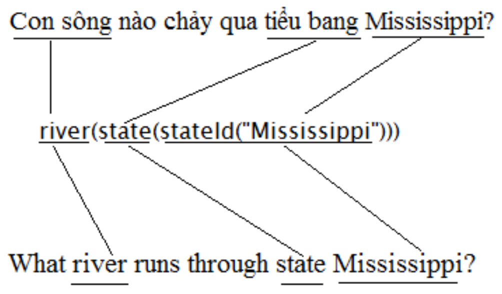
Con sông nào chảy qua tiểu bang Mississippi?
river(state(stateId("Mississippi")))
What river runs through state Mississippi?

Fig. 1. One result of parsing "con sông nào chảy qua tiểu bang missisipi?"

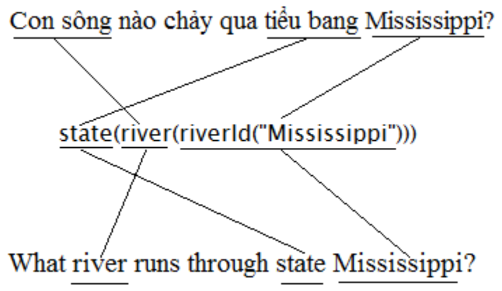
Con sông nào chảy qua tiểu bang Mississippi?
state(river(riverId("Mississippi")))
What river runs through state Mississippi?

Fig. 2. Another result of parsing "con sông nào chảy qua tiểu bang missisipi?"

A method of dynamic learning preference of functions compositions Until now, we have only one feature for describe the composition capacity of two functions. Another one is the feature which describe the compositional preferences of functions. A function is often composed with only a set some other functions. For example, *state* function tends to receive *stateId* function as its argument. In our approach, this feature is learned automatically through the running time of the system.

Suppose $f1$ as a variable p denoting how likely it will be composed with $f2$ to form $f1(f2(arg))$, and this composition appears in a result of semantic parser. By receiving the feedback from the outside world, we may have the answer that the result is correct or not.

If the result is correct In this case, we will increase the value of p by formula 1. It is clear that p will be increased but it is always smaller than M, the maximum value of a feature.

$$p = \frac{(p+1)M}{M+1} \tag{1}$$

If the result is wrong If the user says that he does not expect this answer, p needs to be decreased. Formula 2 make the value of p smaller. p is also supposed to be greater or equal than 0.

$$p = \frac{pM}{M+1} \tag{2}$$

James et al. in [2] also considered a feature indicating which logical symbols are usually composed together. Unfortunately, they did not clearly state the way of computing this kind of feature. If this task is done by manually configuring such feature when we define function, the result would be quite good. But that idea is not really positive with respective to scalability. Large number of functions will overwhelm annotators.

5 Experiment

We use the Geoquery domain for evaluating our system. It consists of a database and Prolog query language for U.S. geographical facts. The corpus has 880 English tokenized queries; each query is paired with a Prolog query. We use Google Translate API to translate the English queries into Vietnamese ones. The translation contains many errors as the result of some word-to-word translations. We then manually correct each query to produce meaningful Vietnamese queries.

Following the experiment in [2], we randomly select 250 queries for training, 250 other queries for testing from the above 880 queries. Our experiment is used to answer the following doubt:

1. What are the effects of dynamic compositional preferences learning?
2. How do two learning approaches of [2] perform in our system?

5.1 What are the effects of dynamic compositional preferences learning?

In this experiment, we run our system with and without dynamic learning of compositional preferences. The experiment result is described in table 1.

"No dynamic learning" row means that we do not use dynamic compositional preferences learning. When calculating the features for function-function compositions, the second feature is the same as the first one. That means they are both computed based on the vector of distance between two tokens.

It is clear from the table that the strategy of dynamic preferences computation improves the accuracy of the system by 4.4% and 3.2% for training data and testing data respectively. We also may wonder that if some set of questions is asked many times by users, how does this strategy perform? The last row of table 1 shows that if we repeatedly run the training data 10 times, then the final result on the training data is surprisingly 62%. In addition, if we run on testing data after running 10 times on training data, the accuracy for testing data is not significantly affected. It decreases by 1.6%.

Algorithm	Training set	Testing set
No dynamic learning	53.6%	51.6%
Dynamic learning	58%	54.8%
Dynamic learning with repetition on training data	62%	53.2%

Table 1. Accuracy of model on traing data and testing data. "No dynamic learning" means that we do not learn compositional preferences during running time of system. "Dynamic learning" means that we apply this kind of learning. in "Dynamic learning with repetition on training data", we run the system 10 times on training data; then testing data is also experimented

5.2 How do two learning approaches of [2] perform in our system?

We implemented two learning mechanisms mentioned in [2]. However, the "Direct Approach" or Binary classification did not successfully learn the weight vector. The learned vector contains negative coefficients. This is because the way we choose the feature of "Token-Function mapping" is simple, just being word matching. We need some semantics features which measure the semantic similarity between words. James et al. in [2] used Wordnet for this purpose. Unfortunately, we do not have Wordnet for Vietnamese. But we had an idea of using word2vec, a tool of Google, to tackle this problem. This is left for our future works.

On the other hand, "Aggressive approach" can learn the weight vector but the result is not improved, around 51%. In future, we need to investigate more about this to take advantage from machine learning.

6 Conclusion

We developed a question answering system using new paradigm of machine learning. The trained data need not to be fully annotated. We only need the questions and the corresponding correct answers. The result at around 55% of accuracy is quite good for Vietnamese semantic parser. However, the current way of calculating features are still quite simple. Improving this function would make our system more powerful. We propose a numbers of possibilities:

1. James et al. in [2] use Wordnet for computing similarity between English words. However, Wordnet is not available for Vietnamese. We plan to use word2vec developed by Google to build a model of similarity between Vietnamese words. After that our system can retrieve the similarity from this model.
2. For function-function mapping, we can first apply syntax parsing for the sentence first. If, for example, two tokens are both in "S" (subject), their two mapped functions are more likely to be combined with each other.

Machine learning methods in [2] does not work well in our system. In the future, we need to investigate more about this.

Currently, we configure the logical functions and database type in one text file. This make the system flexible in terms of database. We can change the database from geometric to others. We can also change the type of database from Prolog to, e.g., MySql. Supporting MySql is also one of our future work.

Bibliography

[1] Ruifang Ge and Raymond J. Mooney. A statistical semantic parser that integrates syntax and semantics. In *Proceedings of the Ninth Conference on Computational Natural Language Learning*, CONLL '05, pages 9–16, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics. URL `http://dl.acm.org/citation.cfm?id=1706543.1706546`.

[2] Clarke James, Goldwasser Dan, Chang Ming-Wei, and Roth Dan. Driving semantic parsing from the world's response. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*, CoNLL '10, pages 18–27, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics. ISBN 978-1-932432-83-1. URL `http://dl.acm.org/citation.cfm?id=1870568.1870571`.

[3] 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, Washington, DC, USA, 2009. IEEE Computer Society. ISBN 978-0-7695-3846-4. doi: 10.1109/KSE.2009.42. URL `http://dx.doi.org/10.1109/KSE.2009.42`.

[4] Le Hong Phuong, Nguyen Thi Minh Huyen, Azim Roussanaly, and Ho Tuong Vinh. Language and automata theory and applications. chapter A Hybrid Approach to Word Segmentation of Vietnamese Texts, pages 240–249. Springer-Verlag, Berlin, Heidelberg, 2008. ISBN 978-3-540-88281-7. doi: 10.1007/978-3-540-88282-4_3. URL `http://dx.doi.org/10.1007/978-3-540-88282-4_23`.

[5] Lappoon R. Tang and Raymond J. Mooney. Using multiple clause constructors in inductive logic programming for semantic parsing. In *Proceedings of the 12th European Conference on Machine Learning*, EMCL '01, pages 466–477, London, UK, UK, 2001. Springer-Verlag. ISBN 3-540-42536-5. URL `http://dl.acm.org/citation.cfm?id=645328.650015`.

[6] Yuk Wah Wong and Raymond J. Mooney. Learning synchronous grammars for semantic parsing with lambda calculus, 2007.

[7] John M. Zelle and Raymond J. Mooney. Learning to parse database queries using inductive logic programming. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence - Volume 2*, AAAI'96, pages 1050–1055. AAAI Press, 1996. ISBN 0-262-51091-X. URL `http://dl.acm.org/citation.cfm?id=1864519.1864543`.

[8] Luke S. Zettlemoyer and Michael Collins. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *In Proceedings of the 21st Conference on Uncertainty in AI*, pages 658–666, 2005.

[9] Luke S. Zettlemoyer and Michael Collins. Online learning of relaxed ccg grammars for parsing to logical form. In *In Proceedings of the 2007 Joint*

Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL-2007, pages 678–687, 2007.