# Semantic parsing for Vietnamese Question Answering System

January 28, 2014

**Abstract**

We apply a learning mechanism into semantic parsing for Vietnamese. This kind of learning reduces the burden of providing supervision. The training data does not need to be carefully annotated. We only need some kind of "Question - Answer" data to train our model.

## 1 Introduction

Semantic parsing is a very important process in Natural Language Processing, helping computers to "understand" natural languages. Generally, semantic parsing converts a natural language sentence into a special representation so that computers can read and process it. We call that special representation as "Computer-understandable language". An effective way to do this task is using learning methods to build a model of relationship between natural language structure and "Computer-understandable language" (CUL) structure. Unfortunately, most of the current approaches require very large amounts of fully annotated data in order to obtain a good model. Annotated data means that for each input sentence of natural language, we need the corresponding result sentence of "Computer-understandable language". The later is almost prepared manually by human, which takes a lot of efforts.

Recently, James et al. 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 "Vietnamese geometric question answering system" (VGQAS). The core of the system is semantic parser.

## 2 Computer-understandable language

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 [1], in our system, the format of "Computer-understandable language" is the composition of pre-defined functions. Each function has only one argument. For example, $longest(river(stateId(\text{"}colorado\text{"})))$ is a CUL sentence, being a result of semantic parsing for "con sông dài nhất chảy qua colorado là gì?" ("What is the longest river run through colarado ?"). There are two steps of building a CUL sentence 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.

## 2.1 Predefined Functions

As mentioned, our CUL uses a set of functions in order to represent and process the meaning of each input sentence. Each function has only one parameter and returns a value of some type. In addition, a function should not accept any argument, but only some 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 required to be defined in a text file. This way of configuration makes our system flexible because if we want to change the data, changing the functions configuration file is enough.

## 2.2 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 a lot of works have dealed with tokenization for Vietnamese with high accuracies such as ... (need citation here).

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 are supposed to be aligned with one function from the predefined ones. In the currently considered example, "con sông", "dài nhất" and "colorado" are mapped to *river*, *longest* and *stateId* functions respectively.

## 2.3 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 $f2$ should agree with the argument type 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 CUL sentence of the form $longest(river(stateId(\text{"}colorado\text{"})))$.

# 3 VGQAS overview

The goal of the system is to answer the questions of users properly. The core of our system is semantic parsing. As presented, given an input sentence $x$, our semantic parser tries to make a mapping $y$ between tokens and functions. Then from the mapped functions, it will search for a composition $z$ of them for final logical form. The problem is that for each $x$, there may be some acceptable $y$; and for each $y$, there are some acceptable $z$. But semantic parser is required to choose only one pair of $y$ and $z$.

Our system will consider all the possible cases and putting scores on relations between $x$ and $y$, $y$ and $z$. After that, it decides the best meaning representation in terms of the total scores of each pair of mapping and composition. More precisely, the prediction function is as follow:

$$\hat{y}, \hat{z} = arg\ max_{y,z}(score(x, y) + score(y, z)) \tag{1}$$

We represent the *score*s by features vectors and weight vector. Let $\theta_1(x, y)$ and $\theta_2(y, z)$ be the feature vectors that represent relations between $x$ and $y$, $y$ and $z$ respectively. $w$ is the feature vector. Equation 1 now become:

$$\hat{y}, \hat{z} = arg\ max_{y,z} w^T(\theta_1(x, y) + \theta_2(y, z)) \tag{2}$$

The feature vectors and weight vector have the same size. In our system, their size are 3. Solving the prediction in equation 2 is done by Linear Programming solver.

Weight vector is obtained by machine learning. We apply two learning methods mentioned [1] which are "Direct approach" and "Aggressive Approach".

## 3.1 Direct Approach

## 3.2 Aggressive Approach

# 4 Semantic parsing model

As mentioned, each parser system has a model for translating a natural language sentence to CUL sentence. In VGQAS, model consists of two parts: Word-Function feature calculator and Function-Function feature calculator.

## 4.1 Word-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 [1], 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 [1].

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 $f_1(f_2)$. The most important note is that returned type of $f_2$ must agree with argument type of $f_1$. The second feature is the distance between the tokens corresponding 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. Let us consider the following example:
"con sông nào chảy qua tiểu bang missisipi?" ("what river runs through state mississippi?").

# References

[1] 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`.