Report for EMNLP project 2

Leqiang Wang
1500012938
attilawlq@pku.edu.cn

Abstract

In this project, we should translate the answer to its questions in english into certain logic form. We only have hundreds of examples a task. This is a complicated problem, especially when trying to solve it without any deep learning method. In perspective on this problem, there are two major issues: formulating the answer's logic form and key phrases mapping. Due to the limit of time and experience of programming, I only tackle these two issues in simple and arbitrary ways.

1 Formulation of Answer's Form

The form of answers is very confusing. There isn't any explicit logical form for answers given for this project. According to my observation, the answers can be generated from the following CFGs:

$$S \longrightarrow func_{answer}$$
$$func \longrightarrow func\_tag(prop, prop, ...prop)$$
$$prop \longrightarrow func|component|const\_name$$
$$|(prop, prop...)$$
$$component \longrightarrow A|B|...$$
$$const\_name \longrightarrow mississippi|newyork...$$
$$func\_tag \longrightarrow place|loc|river...$$

Each function such as "loc", "river" has a fixed number of parameter, which is a list, the component of the list is either a capital letter or a function. The type of each component of each "function" is fixed. The problem is, that there could be function embedded in s function, such a form is not what I can deal with directly. I "flatten" the answer into a form that only allow functions contain capital letter or const names as parameter by recursively move the parameters which is not a letter or const name to the right of the function.

For example, for the following fraction of answer:

$$(A, count(B, (state(B), next_to(B, C), const(C, stateid(iowa))), A))$$

This is a list of *props*, including function count() that has a parameter that is a list of *props*. After being flattened, it is transformed like this:

$$(A, count(B, A), state(B), next_to(B, C), const(C, stateid(iowa)))$$

The *const* is handled in a specialized way, it is treated as a single parameter function $const\_iowa(C)$, and there is a dict that saves what such const function literally look like. For $const\_iowa(C)$, first it is transformed into $const(C, iowa)$ when parsing, then $iowa$ map to literally $stateid(iowa)$ in a const name dictionary.

When transforming the answer to flattened form, those functions with unflat parameter are recorded with their forms of parameters. When translating the question, I first translate the question into the flattened form, then recover it to the standard form from the flattened form.

The recovery methed comes from my observation on answers. When recovering, I scan the whole flattened function list from back. If there is an unflat function, collect all the functions at right side of the current unflat function that "describe" its flat parameters.

For example, to recover the following answer:

$$(highest(A), place(A), loc(A, B), const\_hawaii(B))$$

The *highest()* is an unflat function, it has two parameters, first one is a flat one which is a capital letter while the second parameter is not flat,

which is a subsequent list (or one single function, in sense of unflat parameter, a single function is equal to a list of function with only one element) that describe the parameter. If a function has common parameter, with the target unflat function, in this case any function at right have parameter A describes $highest$. Also if the function have common parameter with any previous collected function, it describes the function. In the example, $loc(A, B)$ describes $highest$, then $const_hawaii$ has parameter B, and it is the subsequent function of loc, so it describes $highest$.

2 Mapping from Phrase to Function

There are two type of functions in flattened form, which are 1-parameter function and 2-parameter function (or may be more, if there is a function with more than 3 flat parameters, it is handled similarly as 2-parameter function).

2.1 Assumptions

1. Each captal letter represent a NP chunck in the sentence.
2. Each 1-parameter function represent a property of a NP chunck, which maps to a phrase(or a single word) in the corresponding NP chunck.
3. Each 2-parameter function represent relation between NP chuncks, which maps to a phrase in certain range between NP chuncks.

2.2 Finding Key Phrase

The phrase maps to one of the function is called "key phrase". To find key phrase, enumerate all the subsequence of sentence in legal range according to the type of function(in NP chunck only or range between NP chuncks). Then scoring each phrase using conditional probability. Score of a $phrase$ for a $func$ is:

$$score = P(func|phrase) * P(phrase|func)$$

The phrase satisfies $P(func|phrase) > 0.5$ and has highest score is to be selected

Note that in a sentence, sometimes there are key phrases overlap each other. Such situation violates the assumptions. Also, sometimes one phase maps to multiple functions. This may make sense but my goal is to determine to series of function using key word mapping, so multimap should be avoided. When conlict occurs, select the choice that has higher $P(func|phrase)$. Keep Finding key phrases iteratively untill there will be no more conflict.

3 Translation Questions

The translating process is kind of reversed operation of the two sections above.

When a sequence of function is find, it is in a flat form of answer. The parameters are NP chuncks, they should be tagged with capital letters. According to my observation, the index of parameter's letter is the index of its first appearance.

For example:

$$(river(www), traverse(www, gg), state(gg), next_to(gg, kk), largest(uyu), state(kk), population(kk, uyu))$$

www appear first, www is A, then in the (www, gg), gg appear second, which is B, then kk is the third, uyu is the fourth.

So the tagged answer is:

$$(river(A), traverse(A, B), state(B), next_to(B, C), largest(D), state(C), population(C, D))$$

Then recover the origin answer from this flattened one.

4 Weekness and Possible Improvement

I think the major reason for errors on valid set is that I doesn't consider enough context information. The scoring method is too simple, if more advanced method is applied for selecting key word, the result will be better.

5 Source Code

The source code files don't contain any processed file. If BerkerlyParse1.7 are intalled in the source file folder, run 'run.sh' in d1 and d2, results and processed files will be generated.