Syntactic Parsing Review

Zhong Wenjie
zvengin@nii.ac.jp

December 27, 2017

1 PCFG

PCFG similar to CFG is defined by $\{\Sigma, R, N, S\}$, where R incoporates a probability to each rule $r, r \in R$. Compared with CFG, PCFG can mitigate ambiguity and generate probability for given sentence.

1.1 Parse with PCFG

As PCFG is an extention of CFG, the algorithm of PCFG is very similar to CFG. Like CFG, we use a table to store intermediate results. The table is a $(n+1)(n+1)$ matrix and we use the upper triangle to store intermediate results. (1)With repsect to the terminal words, for each column j we check if there is any grammar matching the left side of grammar to the word of column j. If matched, we fill the $cell(j-1, j)$ with the left side of matched grammar. (2)With respect to the other $cell(i, j)$ where $i \in \{1, 2, ..., j-2\}$, for k where $k \in \{i+1, ..., j-2\}$ we check if there is any grammar whose right side matches with the combination of $cell(i, k)$ and $cell(k, j)$. Therefore, there may be several combinational options for $cell(i, j)$ and we choose the combination which has highest probability. The probability of each possible combination can be calculated like this:
if rule r: A $\rightarrow$ BC p_r is applied then the probability of this combination is:

$$p = p_r \times cell(i, k) \times cell(k, j). \tag{1}$$

(3)We can calculate the value of each cell by repeating (2) for each column. Finally we get the parse tree at $cell(0, n)$.

1.2 Generate Pobability of Rule

Given a corpora which includes sentences and corresponding parsing trees. The probability of each rule is calculated by the following formula.

$$P(A \rightarrow BC) = \frac{num(A \rightarrow BC)}{\sum_D num(A \rightarrow D)} \tag{2}$$

Usually we estimate the probability of each rule based on Penn Treebank. For each sentence, Penn Treebank has a parsing tree. If there is no available corpora including parsing trees, maybe we can use a parser to generate parsing trees.

1.3 Problem of PCFG

Even though PCFG has mitigated ambiguity to a great extent, there remian two major problems needed to be tackled. One is that PCFG misses structural dependencies between rules because of independence assumption, the other is that PCFG still lacks of sensitivity to lexical dependencies. (1) The first problem refers to the case that several rules can be applied to expanding the non-terminal and each rule has same probability. In this case, how should we select rule? Actually, in English the choice of how a node expands can after all depend on the location of the node in the parse tree. However, unfortunately PCFG is unable to capture such structural depencies as a result of independent assumption. Therefore, parent annotation, in which we annotate each node with its parent in the parse tree, is introduced to solving this problem. (2)The second problem with PCFG is their lack of sensitivity to the words in the parse trees. If PCFG is sensitive to words in parsing tree, it can utilize lexical dependencies to reslove prepositional phrase attachment ambiguity and coordination ambiguity. With respect to prepositional phrase attachment ambiguity, Usually, the position where pp attaches is decided by the rule which has the highest probability. However, this leads to that PCFG will always prefer to a specific rule, more specificly a rule with highest probablity, even though the chosen rule makes sentence's semantic meanless. However, in English some words are usually matched with some specific prepositional phrases. Therefore, there will be a great improvement if lexical depencies are introduced to PCFG.