0.1 Parsers

The main task of parser is to transform natural language sentences to formal and symbolic representation in the form of trees where syntactic analyses is linked to each string (section parse tree?). These parse trees are collectively called as treebank. We use five different parsers, namely

- Chart Parser
- Earley Chart Parser
- Bottom Up Chart Parser
- Top Down Chart Parser
- Recursive Descent Parser(top down,depth first)

Chart parsing is an example of dynamic programming in the sense that we parse a sub part of input, we store the intermediate results and re-use them when appropriate. Intermediate results are stored in chart.As a result, chart parser doesn't do same work multiple times unlike top-down parser. This prevents both backtracking and combinatorial explosion.

Bottom Up Chart parsing starts from the input string and lookup CFG to find the relevant words or phrases in the right hand side of CFG. The parser replaces these with the left hand side of CFG until the whole sentence is reduced to S(root node).These intermediate results are recorded as edges on a chart. As name suggests, it is one of the variant of chart parser.

Top Down Chart parser starts off with a top level goal, which is from root node (S). This is broken down into sub-goals to find different constituents according to our CFG.These are expanded until a terminal is found. Once terminal is found, it is matched with input string.Accordingly it is added as edge to the chart.And the chart is completed once all the input strings are matched. Here chart is nothing but parse tree.

Earley chart parser is a variant of chart parser and it is based on top down parsing. But it deals more efficiently with matching against input strings by considering both top down and bottom up strategy.

Recursive descent parsing is as same as top down parsing but without using chart to store intermediate results.It executes same work multiple times.

All the five parsers generate similar parse trees in our application because of the naivety of our CFG.Due to this, we consider only chart parser. All the above mentioned parsers are implemented using NLTK(cite?)

0.2 Generators

Once we obtain treebank, we use them to develop PCFG(section PCFG). Thats how the system learns about corpus and we give this PCFG along with weather conditions as input to the generator. The generator will give out the sentence for that particular weather condition according to PCFG . We have implemented five generating algorithms:

0.2.1 ProbGen Generator

In this method, the generator chooses between the rules in PCFG according to their probabilities randomly.

For example: If PCFG has two rules A $->$ B and A $->$ C with probabilities 0.7 and 0.3, then ProbGen chooses rule (1) 7 times out of 10 and rule(2) 3 times out of ten.

0.2.2 Greedy Generator

Greedy generator will choose the production with maximum probability at each node according to PCFG.

0.2.3 Viterbi Generator

Unlike greedy generator, viterbi generator will choose the production with maximum probability along its path from root node(S)

0.2.4 Random generator

Random generator chooses the production randomly irrespective of its probabilities.

0.2.5 n-gram Generator

We have implemented n-gram generator with order 3 with words being each unit.(cite?)

References