NLU Assignment 3: Parsing

Kapil Pathak

Abstract

This report consists of details of the implementation CYK parser discussed in the class.

1 Introduction

In this assignment, a set grammar rules has been created with their probabilities calculated independently from the corpus given in nltk library. The corpus has been divided into train and test set with 80-20 ratio. and the rules were stored according to training set.

2 Parsing

In the languages such as English, Hindi etc, the grammar and the formation of the sentence follow certain rules which are universally accepted. These rules help us to understand the meaning of the sentences. These rules also play important role in various examples of part of speech etc. so that deeper understanding of the sentence will be comprehended. But because of the richness of these languages, it becomes difficult to maintain proper meaning of the sentence. In such cases, even human level accuracy is also low. Such an example can be given as:

Hence the ambiguities have been resulted while parsing. These ambiguities are of various types such as prepositional phrases, particle vs preposition, complement structures, coordination scope etc. Figure 1 explains the different parsed structures for the same sentence.

3 Context Free Grammar

A context free grammar consists of a set of rules with additional set of non-terminal symbols, set of terminal symbols and a start symbol S. A set of non-terminal symbols generally contains all grammar propositions while the set of terminal symbols contain all the words have been occurred into

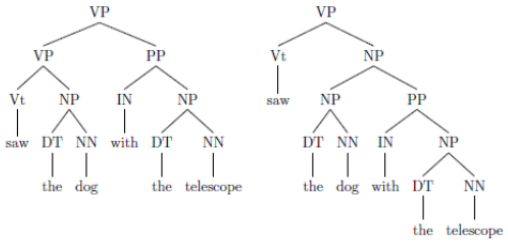

Figure 1: Different parsed structures for the same sentence, source:http://www.cs.columbia.edu/ mcollins/courses/nlp2011/notes/pcf

training set sentences in general. A context-free grammar (CFG) is a 4-tuple $G = (N, \Sigma, R, S)$ where:

- N is a finite set of non-terminal symbols.
- Σ is a finite set of terminal symbols
- R is a finite set of rules of the form $X \rightarrow Y1Y2...Yn$, where $X \in N, n \geq 0$,and $Y_i \in (N \cup \Sigma)$ for i = 1 . . . n.
- $S \in N$ is a distinguished start symbol.

All these machinery in the hand give multiple possible parsing structure. Hence choosing the best one always have been a challenge. Apart from it, there are two problems to solve in this task. Firstly, all possible parsing structures have common broader structures. Hence one needs to avoid repeated work. Figure 2 explains it. Secondly, one need to choose the best one.

4 Probabilistic CKY Algorithm

Mathematically PCKY algorithm can be described as:

- $G = (\Sigma, N, S, R, P)$
- T is a set of terminal symbols

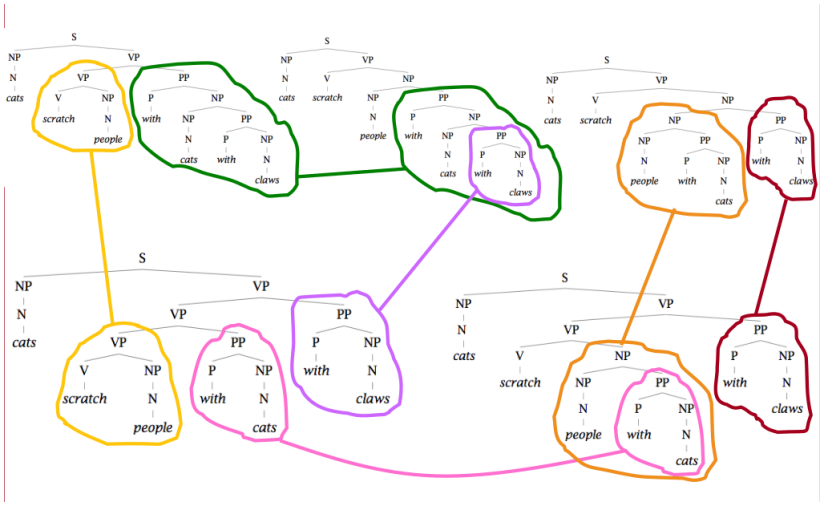

Figure 2: Repeated blocks in parsing, source:Jurafsky lectures

- N is a set of nonterminal symbols
- S is the start symbol ($S \in N$)
- R is a set of rules/productions of the form $X \rightarrow \gamma$
- P is a probability function such that $P : R \rightarrow [0, 1]$ and $\forall X \in N, \sum_{X \rightarrow \gamma \in R} P(X \rightarrow \gamma) = 1$

In this algorithm, we first find all rules of the form unary, binary etc. The we remove all unary forms which come into midway into the parsing tree with collapsed unary form. Then we find Chomsky Normal Form so that all nodes will have maximum 2 children. All such rules have been added to common production object.
For all these rules, probabilities have been calculated by maintaining counts of rules for which parent node is common. And then find a probability for a particular rules by dividing frequency of that rule by counts for parent node. All these rules have been saved to the file G.cfg in github repository.

5 Penn Tree Bank dataset

For this assignment, Penn Tree Bank dataset has been used. This dataset contains a labelled dataset where it contains parsed outputs as well. Due to limited examples in the datset, a smoothing procedure is used while calculating probability for each rule. But from observation point of view for short sentences, it didn't seem to add much difference.

6 Implementation

The submitted codes consists of two implementations. Both implementations follow the same psuedo code given in the lecture presentation. In the first implementation, all productions are iterated through the keys as the pseudo code given in the lecture presentation requires order n^3 processing time. But as keys in python are iterated randomly, first implementation is not giving consistent results. To mitigate this factor, the second implementation consists of a loop over labelled list keys, but it again requires order n^3 processing time. Hence for longer sentences it takes considerable amount of time to get see the results. One of parsed example for the sentence "I went home" is shown below.

```
(S
  (NP-SBJ (PRP I))
(VP
  (VBD went)
(NP
  (NN home))))
```

Figure 3: Parsing on short sentence

7 Acknowledgement

Here I acknowledge following references through which the above implementations have been benefited.

- https://github.com/rdorado

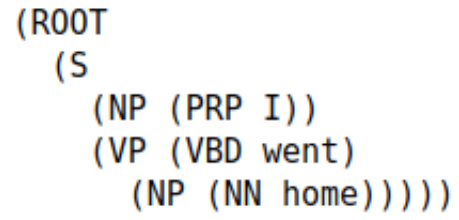

Figure 4: Parsing on short sentence by Stanford parser

- http://www.cs.columbia.edu/ mcollins/courses/nlp2011/notes/pcfgs.pdf
- https://raw.githubusercontent.com/mmera/ckyParser/master/cky.py