Implementation of Constituency Parsing using PCFG and CKY Algorithm

Ankur Debnath
M.tech (EE)
ankurdebnath@iisc.ac.in

Abstract

In this task, a Probabilistic Context Free Grammar based Constituency Parsing was performed. A Viterby-CKY n-best parser was used for implementation. The dataset used for creating grammar as well as training and testing is the Penn Treebank Wall Street Journal dataset made available by the NLTK module in Python. Finally the parser was evaluated on the basis of Precision, Recall and F1-Score which was found to be 0.6390 .

1 Introduction

Parsing is the task of breaking a data block into smaller chunks by following a set of rules (grammar), so that it can be more easily interpreted, managed, or transmitted by a computer. In particular, constituency parsing aims to extract a constituency-based parse tree from a sentence that represents its syntactic structure according to a phrase structure grammar.Recent approaches convert the parse tree into a sequence following a depth-first traversal in order to be able to apply sequence-to-sequence models to it.

In this experiment, 2 tasks were performed : Implementation of CKY parser on the penn treebank dataset and evaluation comparision with standard online parsers available. While performing this experiment, the grammar rules were generated using Probabilistic Context-Free Grammar(PCFG) in Chomsky Normal Form(CNF)

2 Task 1 : Implementation and Evaluation

2.1 Grammar generation

The grammar rules to be used by the parser needs to be generated upfront. For this a 80-20 split was used on the dataset. The PCFG probailities were learned from the train trees using ML estimates and converted finally to CNF. A sample rule is given by

NNP -> Carballo [0.000086]

2.2 Smoothing

Unknown word problem is dealt by purposefully plugging-in a specific word - ”<UNK>” in the training corpus.After plugging-in, the lexicon contains probability of the word ”<UNK>” and thus when an unknown word is encountered in the test corpus, the probability of the ”<UNK>” token is used. While generating the grammar Good-Turing smoothing was used which initially recognizes each new word as ”<UNK>”, then looping over the list of tokens from beginning to start, each time a new token appears, it was replaced with ”<UNK>”.

2.3 CKY Parsing

It is a bottom-up dynamic programming algorithm to build a collection of derivations of sub-strings from grammar symbols. It can only recognize languages defined by PCFGs in CNF. Initialization is done by assigning to each non-terminal from the sentence with the terminal from the grammar; if there is more than 1 terminal which results in current non-terminal then both of them are remembered. As a result, imagined triangular table has its bottom filled with non-terminals right above the starting terminals.

Optimization part is a structure of 3 nested for loops: for each span, for each beginning and ending of sentence (ending is defined to be beginning + span) while looking for the non-terminal, which is the root of the most probable tree, which dominates the children in the range from the beginning to ending. The maximum non-terminal is found using a split-point from the range beginning + 1

Evaluation Parameters	Value
Coverage	67.24%
Average Precision (Parsed)	0.9504
Average Recall (Parsed)	0.9504
Average F1-Score (Parsed)	0.9504
Average Precision (Total)	0.6390
Average Recall (Total)	0.6390
Average F1-Score (Total)	0.6390

Table 1: Evaluation Parameters

and endŁŁ1 recursively to find all possible words combinations.

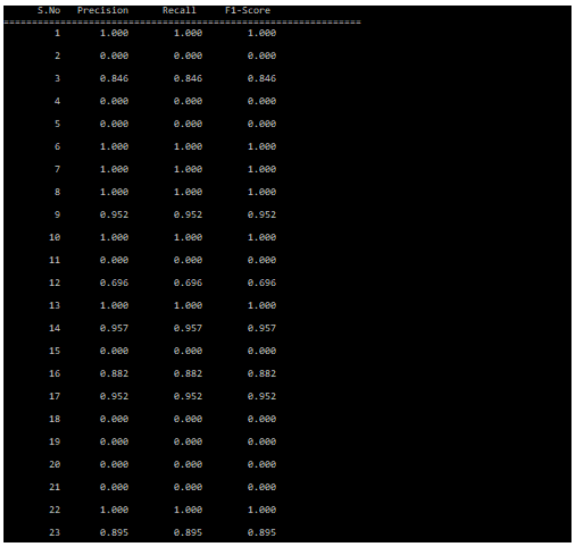
This Algorithm generates the best parse tree of the imput sentence with the highest probability. In this experiment, a Viterbi-CKY n-best parser war used provided by NLTK which generates the best probabilistic parse tree of the input sentence.

2.4 Evaluation

The grammar rules were trained on the training set while each sample from the test, which includes raw text and parse tree were treated as gold standard and were compared against the parses generated by our parser and finally the evaluation was done between the two trees for each sample in terms of Precision, Recall and F1-Score. Table 1 shows the results.

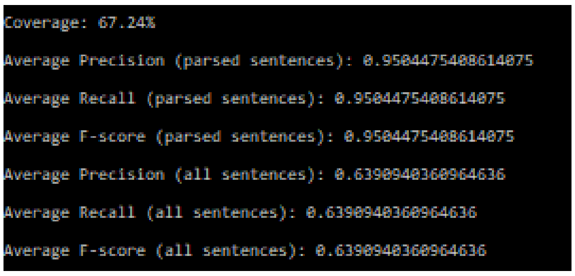
As some parses were not of the same length as the gold standard they were considered as unparsed and a total of 67.24% coverage was obtained on the test set. Ideally a good parser has an average score 0.6-0.75 F1-score. The parser in our case yielded 0.6390 F1-Score on the testset.

Figure 1: Evaluation Scores on the testset



```
      S.No   Precision       Recall     F1-Score
================================================================
         1       1.000        1.000        1.000
         2       0.000        0.000        0.000
         3       0.846        0.846        0.846
         4       0.000        0.000        0.000
         5       0.000        0.000        0.000
         6       1.000        1.000        1.000
         7       1.000        1.000        1.000
         8       1.000        1.000        1.000
         9       0.952        0.952        0.952
        10       1.000        1.000        1.000
        11       0.000        0.000        0.000
        12       0.696        0.696        0.696
        13       1.000        1.000        1.000
        14       0.957        0.957        0.957
        15       0.000        0.000        0.000
        16       0.882        0.882        0.882
        17       0.952        0.952        0.952
        18       0.000        0.000        0.000
        19       0.000        0.000        0.000
        20       0.000        0.000        0.000
        21       0.000        0.000        0.000
        22       1.000        1.000        1.000
        23       0.895        0.895        0.895
```

Figure 2: Average of Evaluation Scores on the testset

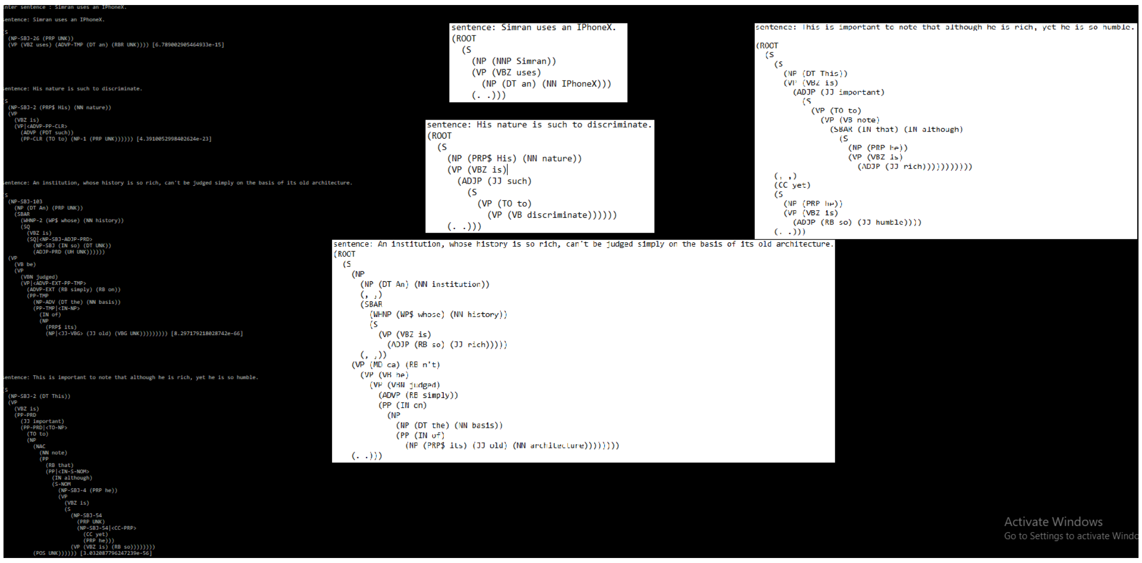


```
Coverage: 67.24%

Average Precision (parsed sentences): 0.9504475408614075

Average Recall (parsed sentences): 0.9504475408614075

Average F-score (parsed sentences): 0.9504475408614075

Average Precision (all sentences): 0.6390940360964636

Average Recall (all sentences): 0.6390940360964636

Average F-score (all sentences): 0.6390940360964636
```

3 Comparision

The comparison of the parser was done against 2 standard available parsers from Stanford[1] and Berkeley[2] with 2 short and 2 long sentences. The results are shown in the figure 3. It is clear that there are many errors especially with the unknown words for the grammar used. Experiments were performed by tweaking the probaibilities of <UNK> tokens but not much success was achieved regarding the parsing when compared using F1-Scores on the 4 test sentences. Although a slight improvement of F1-score was observed when Exponential Smoothing was used over Good-Turing Smoothing for the <UNK> tokens.

Figure 3: Comparision



```
nter sentence : Simran uses an IPhoneX.

entence: Simran uses an IPhoneX.

(S
  (NP-SBJ-26 (PRP UNK))
  (VP (VBZ uses) (ADVP-TMP (DT an) (RBR UNK)))) [6.789002905464933e-15]



entence: His nature is such to discriminate.

(S
  (NP-SBJ-2 (PRP$ His) (NN nature))
  (VP
    (VBZ is)
    (VP|<ADVP-PP-CLR>
      (ADVP (PDT such))
      (PP-CLR (TO to) (NP-1 (PRP UNK)))))) [4.3910052998402624e-23]



entence: An institution, whose history is so rich, can't be judged simply on the basis of its old architecture.

(S
  (NP-SBJ-103
    (NP (DT An) (PRP UNK))
    (SBAR
      (WHNP-2 (WP$ whose) (NN history))
      (SQ
        (VBZ is)
        (SQ|<NP-SBJ-ADJP-PRD>
          (NP-SBJ (IN so) (DT UNK))
          (ADJP-PRD (UH UNK))))))
  (VP
    (VB be)
    (VP
      (VBN judged)
      (VP|<ADVP-EXT-PP-TMP>
        (ADVP-EXT (RB simply) (RB on))
        (PP-TMP
          (NP-ADV (DT the) (NN basis))
          (PP-TMP|<IN-NP>
            (IN of)
            (NP
              (PRP$ its)
              (NP|<JJ-VBG> (JJ old) (VBG UNK))))))))) [8.297179218028742e-66]



entence: This is important to note that although he is rich, yet he is so humble.

(S
  (NP-SBJ-2 (DT This))
  (VP
    (VBZ is)
    (PP-PRD
      (JJ important)
      (PP-PRD|<TO-NP>
        (TO to)
        (NP
          (NAC
            (NN note)
            (PP
              (RB that)
              (PP|<IN-S-NOM>
                (IN although)
                (S-NOM
                  (NP-SBJ-4 (PRP he))
                  (VP
                    (VBZ is)
                    (S
                      (NP-SBJ-54
                        (PRP UNK)
                        (NP-SBJ-54|<CC-PRP>
                          (CC yet)
                          (PRP he)))
                      (VP (VBZ is) (RB so))))))))
          (POS UNK)))))) [3.032087796247239e-56]
```

```
sentence: Simran uses an IPhoneX.
(ROOT
  (S
    (NP (NNP Simran))
    (VP (VBZ uses)
      (NP (DT an) (NN IPhoneX)))
    (. .)))
```

```
sentence: His nature is such to discriminate.
(ROOT
  (S
    (NP (PRP$ His) (NN nature))
    (VP (VBZ is)
      (ADJP (JJ such)
        (S
          (VP (TO to)
            (VP (VB discriminate))))))
    (. .)))
```

```
sentence: This is important to note that although he is rich, yet he is so humble.

(ROOT
  (S
    (S
      (NP (DT This))
      (VP (VBZ is)
        (ADJP (JJ important)
          (S
            (VP (TO to)
              (VP (VB note)
                (SBAR (IN that) (IN although)
                  (S
                    (NP (PRP he))
                    (VP (VBZ is)
                      (ADJP (JJ rich))))))))))))
    (, ,)
    (CC yet)
    (S
      (NP (PRP he))
      (VP (VBZ is)
        (ADJP (RB so) (JJ humble))))
    (. .)))
```

```
sentence: An institution, whose history is so rich, can't be judged simply on the basis of its old architecture.
(ROOT
  (S
    (NP
      (NP (DT An) (NN institution))
      (, ,)
      (SBAR
        (WHNP (WP$ whose) (NN history))
        (S
          (VP (VBZ is)
            (ADJP (RB so) (JJ rich)))))
      (, ,))
    (VP (MD ca) (RB n't)
      (VP (VB be)
        (VP (VBN judged)
          (ADVP (RB simply))
          (PP (IN on)
            (NP
              (NP (DT the) (NN basis))
              (PP (IN of)
                (NP (PRP$ its) (JJ old) (NN architecture))))))))
    (. .)))
```

Conclusion

The experiments replicated the benchmark F1-scores of CKY Parsers with PCFG grammar rules using Good-Turing Smoothing. But when compared with the standard online available parsers and test sentences out of the training dataset, the performance reduced drastically on longer sentences as compared to shorter ones. This shows the limitation of such dynamic programming approaches to learn the underlying structure properly and shows the need for much generic decomposition techniques such as LSTMs and other deep architectures and treating the sentences as sequential input.

References

1.Stanford Parser
2. Berkeley Parser
3. GitHub Link