Assignment 3: E1-246
PCFGs and CKY : Constituency Parsing

Aadesh Magare
aadeshmagare@iisc.ac.in

Abstract

This is a report for assignment 3 of E1-246 (NLU) course. It contains experimental details about the task of PCFGs and constituency parsing. PCFG is implemented with the penn treebank dataset. CKY algorithm is used for constituency parsing along with smoothing mechanisms.

1 Datasets:

The Penn Treebank dataset provided by nltk (TreeBank) was used for the task. It consits of 200 files which were split into training (160) and test (40) datasets. Averaged labelled precision, recall and F1 score are used as evaluation metrics.

2 Model:

Parsed trees from the training data are used to extract productions. ML estimate (relative frequencies) on these productions gives probabilistic productions which are used to construct PCFG. The productions are first converted into CNF (CNF) form and unary productions are collapsed before constructing PCFG.

The CKY algorithm (CKY) is implemented for constituency parsing using the generated PCFG. During validation time, out of vocabulary words i.e. words not covered by the PCFG productions are added into the grammar as noun (NN) productions and while probabilities for other productions are smoothed out. OOV words during testing are treated as <UNK> tokens.

3 Evaluation:

3.1 Add One Smoothing:

Simple add one smoothing is implemented where unseen word productions are given count of 1 and related probabilities are adjusted accordingly.
The model evaluated with add one smoothing achieves the results displayed in table 1

Metric	Value
Precision	32.36
Recall	40.93
F1	36.14
Tag Accuracy	77.37

Table 1: Evaluation results: Add-1 smoothing

3.2 Interpolation Smoothing:

Interpolation smoothing with uniform weights is implemented where a unseen word production is given probability equal to average of probabilities for that non-terminal.
Here we are using interpolation factor of 1 / n where n is no of productions for the non terminal.
The model evaluated with add one smoothing achieves the results displayed in table 2

Metric	Value
Precision	33.45
Recall	42.21
F1	37.33
Tag Accuracy	76.79

Table 2: Evaluation results: Interpolation smoothing

4 Comparison:

The best parser from this experiment is compared against (Parser) to produce few error instances.

4.1 Short Sentences:

1. Programming is an insanely tough task
2. Hard work is hard to do

Analysis:
1. The parser fails to successfully parse this short sentence where as online available parsers are able to do so.

'Programming' and 'insanely' are OOV words and converted to <UNK>. Further 'insanely' is correctly mapped to 'RB' but 'Programming' is wrongly mapped to 'PRP'. This can be fixed with increasing probability of most likely to occur productions, like the production NN →<UNK>

2. The parser fails for the word 'Hard', it being a OOV word is mapped to token <UNK> which is further treated as 'DT' instead of 'JJ'. This occurs since all productions generating <UNK> are given same probability and can be fixed with increasing probability of most likely ones, like the production JJ →<UNK>

4.2 Long Sentences:

1. The roadway continues east as the limited-access expressway Niagara Regional Road 420, which was transferred to the jurisdiction of the Regional Municipality of Niagara in 1998.

2. Board of directors approved the appointment of new CEO regardless of constant opposition from the employees and shareholders resulting in an environment of distrust and anger within the organization.

Analysis:
1. The parser cannot parse this sentence as most of the words are OOV. All of them are converted to <UNK>, thus at each parsing step one of <UNK< generating production is selected regardless of token type. Due to large portion of <UNK> tokens, the parse tree is not generated properly. To fix this number of OOV words from the sentence need to be reduced by expanding the vocabulary.

2. Here a wrong production is selected involving <UNK> token ('distrust') , which further makes the wrong parse tree. decreasing the probability of this production or including the word in vocabulary to avoid <UNK> will fix the parsing.

5 Repository:

The project is available at (GitHub) which contains implementation and usage details.
The trained PCFG can be downloaded from (Url)

References

CKY. Cky algorithm.

CNF. Cnf: Chomsky normal form.

GitHub. Pcfg and cky - nlu.

Parser. Stanford parser.

TreeBank. Nltk: Corpus readers.

Url. Download pcfg.