Homework 4: Parsing
CSCI 662
Submitted by : Karan Singla

Question 1.
Grammer has 752 rules. (excluding non-terminals) : Unary and binary rules

Most frequent rule : PUNC -> . # 346
Best binary rule : PP -> IN NP_NNP # 239

Usage : python code/cfg/cfg_count_v1.py train.trees.pre.unk > rule_counts

NOTE : if you want to see CFG with probabilities then modify line 142 and set flag = 1, else it prints the counts

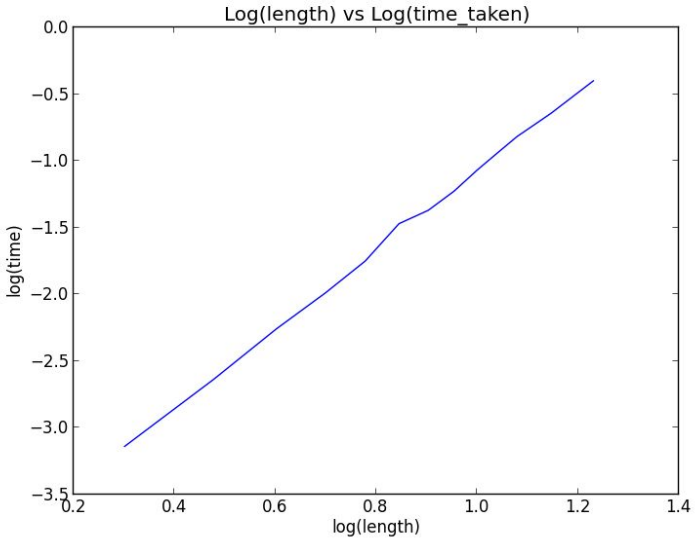
Question 2.
Input : The flight should be eleven a.m tomorrow .

Output : (TOP (FRAG_NP (NP* (NP (NP* (JJ The) (NN flight)) (NN should)) (PP (IN be) (NP (CD eleven) (RB a.m)))) (NP_NN tomorrow)) (PUNC .))

Usage : python code/cyk/cyk_parser.py rule_counts dev.strings > dev.parses

NOTE : It will produce "error" for a sentence it's not able to make a tree for it.

Question 3.



Question 4.

Parser didn't produce any tree for 3 sentences. : Fails to find root.

```
Karans-MacBook-Pro:hw4 Singla$ python postprocess.py dev.parses > dev.parses.post
Karans-MacBook-Pro:hw4 Singla$ python evalb.py dev.parses.post dev.trees
dev.parses.post 457 brackets
dev.trees       474 brackets
matching        404 brackets
precision       0.884026258206
recall  0.852320675105
F1      0.867883995704
Karans-MacBook-Pro:hw4 Singla$
Karans-MacBook-Pro:hw4 Singla$
```

Question 5.

####### Modification 1 ######
Technique to handle word sparsity:

Problem : there are some words which are not there in the training data as it was really small.

Solution : Replace words by word_cluster ID's

Detail :

I trained a Word2Vec[1] model using mono-lingual data for English. Then these word vectors were clustered using KNN clusters to vocab_length / 5 number of clusters.

All the words are replaced in the training data using the mappings obtained for each word to the cluster ID.

Data & Implementation:
Data for training Word2Vec was obtained from statmt.org. It has 5536576 sentences and 87,536 unique vocabulary words. Vector space model was trained using 10 epochs of stochastic gradient descent, and use a context window of size 5. I also considered words which have frequency greater than 5 in the data.

For KNN, I used Sklearn library in Python.

Results after this for CYK parser: [it missed 3 sentences]

```
Karans-MacBook-Pro:hw4 Singla$
Karans-MacBook-Pro:hw4 Singla$ python evalb.py dev.id.parses.post dev.trees
dev.id.parses.post      444 brackets
dev.trees       474 brackets
matching        393 brackets
precision       0.885135135135
recall  0.829113924051
F1      0.856209150327
Karans-MacBook-Pro:hw4 Singla$
```

My intuition was that it should reduce sparsity but training data was not big enough to show that difference. This was because now each lexicon was represented using a word cluster ID. Using a bigger training data will surely increase the recall with this method, even if precision can reduce due to more ambiguity being introduced.

Usage:
‘’’ generate word-cluster mapping file : generates en.map which will be used by next step‘’’
python code/classification.py <en-raw-data>

‘’’ generates rules where some words are replaced by cluster ID’s ‘’’
python code/cfg/cfg_count_id.py train.trees.pre.unk > rule_counts.id

‘’’ generates parsed outputs for dev.string ‘’’
python code/cyk/cyk_parser.py rule_counts.id dev.strings > dev.parses

####### Modification 2 ######

Vertical of 1 & horizontal markovization of 2 of input trees using NLTK:

```
Karans-MacBook-Pro:hw4 Singla$ python evalb.py dev.parses.post dev.trees
dev.parses.post 452 brackets
dev.trees       474 brackets
matching        398 brackets
precision       0.880530973451
recall  0.839662447257
F1      0.859611231102
Karans-MacBook-Pro:hw4 Singla$
```

In this approach too, parser needs to be trained on more/bigger data treebank data.

Modification 3
In the basic cyk parsing I added 4 rules manually to rule count

1 UN-RULE NN M

1 UN-RULE PP I
1 UN-RULE NN A
1 UN-RULE NNP me

Which increased accuracy to

```
Karans-MacBook-Pro:hw4 Singla$ python evalb.py dev.parses.post dev.trees
dev.parses.post 474 brackets
dev.trees       474 brackets
matching        418 brackets
precision       0.881856540084
recall  0.881856540084
F1      0.881856540084
Karans-MacBook-Pro:hw4 Singla$
```

Usage:

‘’’ generates rules where some words are replaced by cluster ID’s ‘’’
python code/cfg/cfg_count_markov.py train.trees.pre.unk > rule_counts.markov

#NOTE : one change the amount of horizontal and vertical markovization by changing parameter in line 126 of code/cfg/cfg_count_markov.py
‘’’ generates parsed outputs for dev.string ‘’’
python code/cyk/cyk_parser_markov.py rule_counts.markov dev.strings > dev.parses

Question 6 : Best accuracy is:

```
Karans-MacBook-Pro:hw4 Singla$ python evalb.py dev.parses.post dev.trees
dev.parses.post 474 brackets
dev.trees       474 brackets
matching        418 brackets
precision       0.881856540084
recall  0.881856540084
F1      0.881856540084
Karans-MacBook-Pro:hw4 Singla$
```

Modification 3 can also be applied to Modification 2,3 to gain better results. But it will be still on dev. Increasing train data will surely help.

References:

1. Mikolov, Tomas, et al. "Efficient estimation of word representations in vector space." *arXiv preprint arXiv:1301.3781* (2013).
2. http://www.statmt.org/wmt14/translation-task.html