Q1

Total number of rules: 723
Most frequent rule: (PUNC .) , it occurs 346 times.

Q2

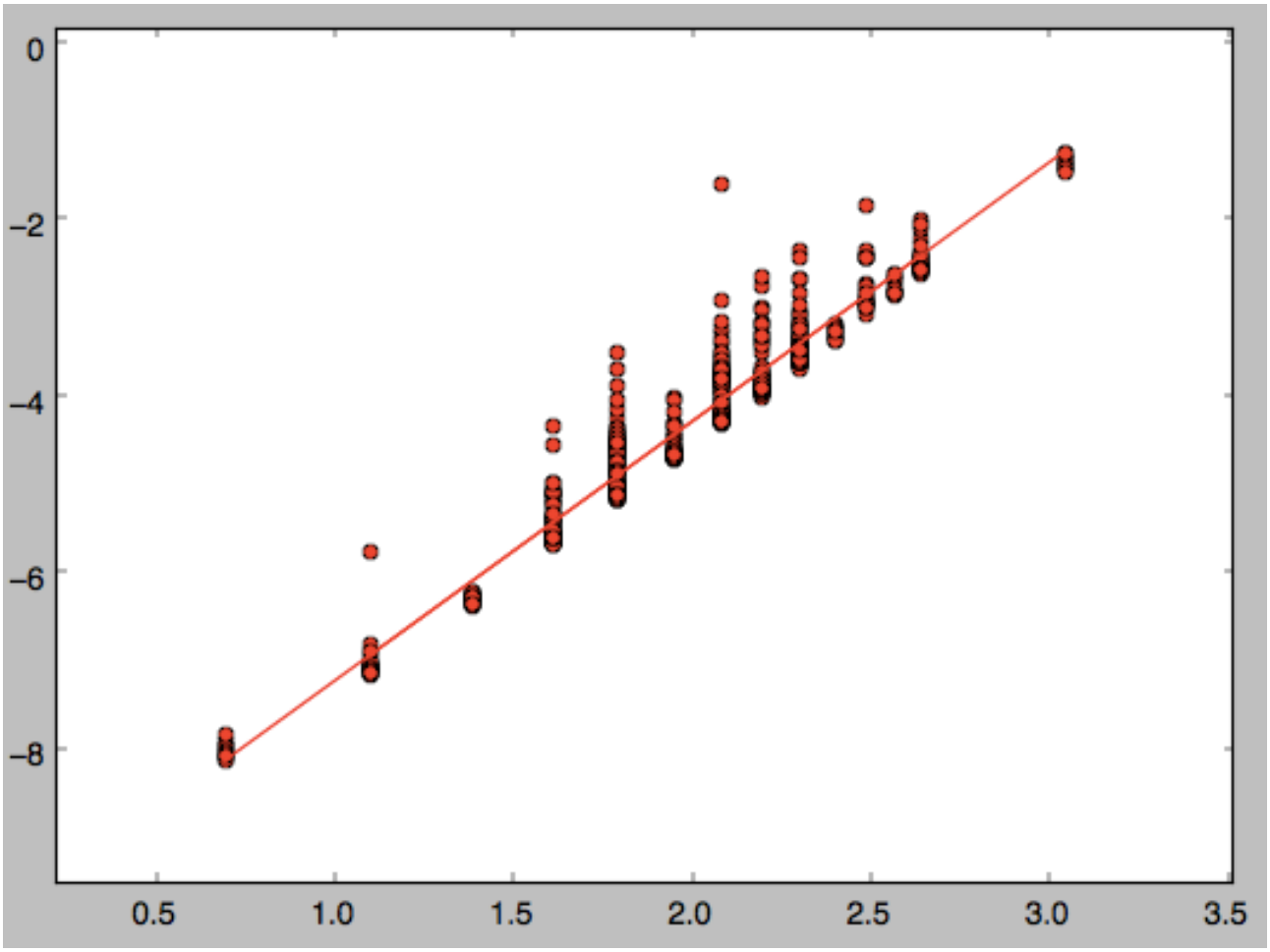
```
-16.0492451274

(TOP (S (NP (DT The) (NN flight)) (VP (MD should) (VP (VB be) (NP (NP*
(CD eleven) (RB a.m)) (NN tomorrow))))) (PUNC .))
```

Q3

```
>>> from scipy.optimize import curve_fit
>>> def func(x, a, b):
>>>    return a * x + b
>>> popt, pcov = curve_fit(func, X, Y)
>>> plt.plot(X, func(X, *popt), 'r-', label="Fitted Curve")
>>> plt.show()
>>> print popt[0]
>>> 2.93750965
```

Yes, k is approximately equal to 3. This is because ViterbiCKY has O(n^3) complexity (it has 3 O(n) nested cycles in the main part of it).

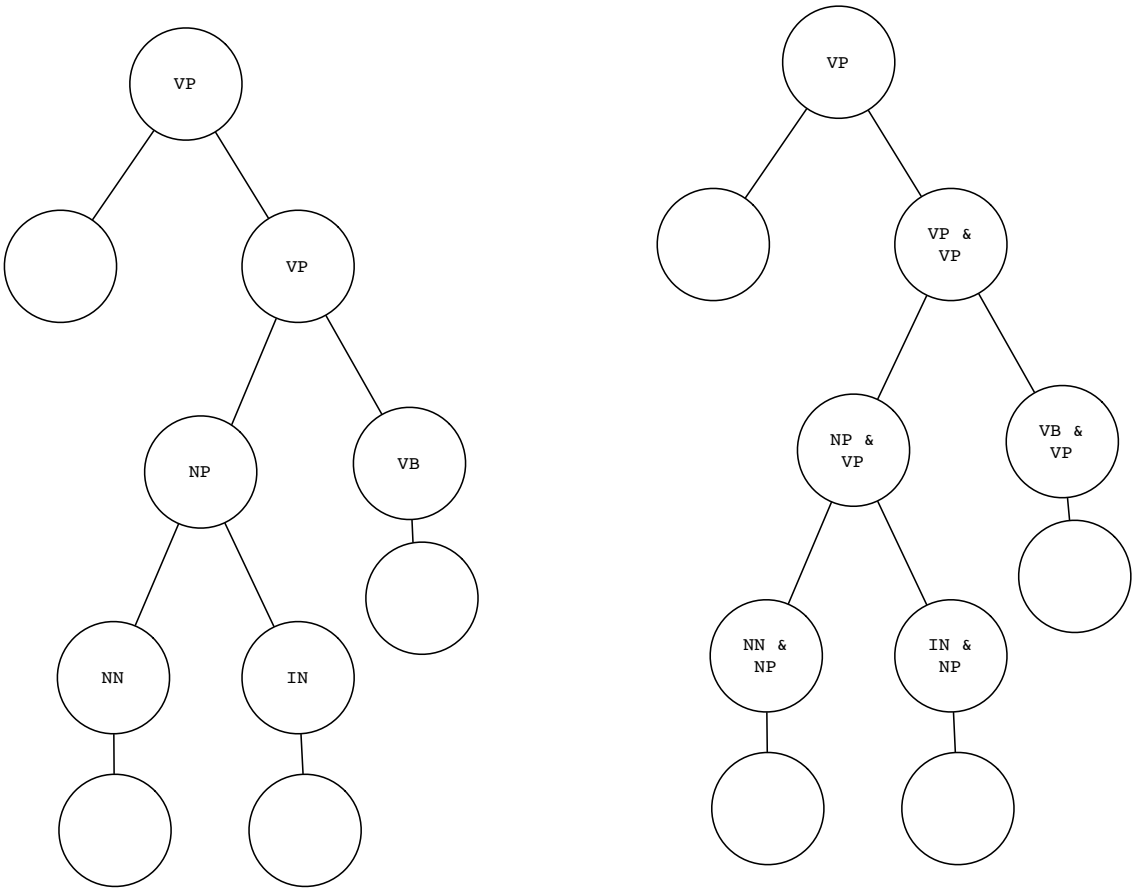


Q4

```
dev.parses.post  454 brackets
hw5/dev.trees    474 brackets
matching   416 brackets
precision  0.916299559471
recall     0.877637130802
F1    0.896551724138
```

Q5

1. Vertical Markovization - the idea is to take into account parent nodes of non-leaf nodes in training trees:



And generate new rules with added annotations, such that:

VP & VP -> NP & VP, VB & VP
NP & VP -> NN & NP, IN & NP

…

Then we can estimate probabilities in grammar using the same “count and divide” method as before.

After parsing a string, we should also remove annotations from parse tree.

2. For horizontal markovization the idea is similar, but instead of adding parent nodes to annotation we need to add sibling nodes when estimate rule probabilities.
3. Here I modified grammar learning procedure to use Lidstone Smoothing.

1-2 modifications decreased F1 score on dev data, I think this is because they both introduced new rules (1546) and increased sparsity of model (higher precision of VM+SMOOTH may indicate this), while the size of training set is remained too small to get good estimation of larger number of model parameters.

Lidstone Smoothing with λ=0.7 slightly improved F1 score:

model	F-score	Precision	Recall
VM + HM + SMOOTH	0.760705289673	0.637130801688	0.637130801688
VM + SMOOTH	0.832167832168	0.9296875	0.753164556962
HM + SMOOTH	0.795294117647	0.898936170213	0.713080168776
SMOOTH	0.896551724138	0.916299559471	0.877637130802
NO Modifications	0.893433799785	0.912087912088	0.87552742616

Q6

If I merge dev and train data into new training set:

```
test.parses.post 462 brackets
hw5/test.trees   471 brackets
matching   424 brackets
precision  0.917748917749
recall     0.900212314225
F1    0.908896034298
```

Original training set:

```
test.parses.post 451 brackets
hw5/test.trees   471 brackets
matching   405 brackets
precision  0.89800443459
recall     0.859872611465
F1    0.87852494577
```

Code:

- hw5/learn_grammar.py - grammar probabilities estimator
- hw5/grammar.py - class for manipulating with grammar
- hw5/parser.py - ViterbiCKY parser
- hw5/parse.py - this script runs parser

Pipelines for annotating and de-annotating data:

- hw5/post_hmark.py
- hw5/post_vmark.py
- hw5/pre_hmark.py
- hw5/pre_vmark.py

./run.sh - this runs the whole pipeline and outputs evaluation results

Final output is stored in folder output/