NLP Homework 3

Itay Itzhak 305685877

Eytan Chamovitz 203486550

Question 1

(a) In code

Question 2

(a) In code
(b) In code. The accuracy on the development set is 91.061%

Question 3

(a) In code
(b) Our pruning policy was devised of two logics.
The first one is that the only possible tags for a word are tags that have been seen in training – since it's emission probability would be zero so we cannot consider those tags which.

$$S_k = \{tag \mid (sentence[k], tag) \in e\_counts\}$$

The second one is that some words occur many times in the text and have known tag (e.g. '.' and its tag '.'). So, we looked for the popular words in train data which have known tag and in test when encountering those words gave them a tag set that is comprised of only that tag.
Optimal lambdas are:
lambda 1 0.7777702222222224, lambda2 0.2222207777777778, lambda 3 8.999999999842467e-06

(c) HMM DEV accuracy: 0.950170750555

(d)

*keep in mind the ancient Chinese game 'Go' while reading this example.

Train sample:
x_1 =(Go Home) $y_1 = (V, NN)$
x_2 =(Go green) $y_2 = (V, JJ)$
x_3 =(Go is fun) $y_3 = (NN, V, JJ)$
x_4 =(I am great) $y_3 = (PRP, PRP, JJ)$

Test Sentence s:
Go is great
(NN, V, JJ)

Q parameters: (assume no smoothing for simplicity)

$$q(V \mid *,*) = \lambda_1 * \frac{c(*,*,V)}{c(*,*)} + \lambda_2 * \frac{c(*,*)}{c(*)} \lambda_3 * \frac{c(*)}{M} = \frac{c(*,*,V)}{c(*,*)} = \frac{1}{2}$$

$$q(V|*,V) = \lambda_1 * \frac{c(*,V,V)}{c(*,V)} + \lambda_2 * \frac{c(*,V)}{c(V)} \lambda_3 * \frac{c(V)}{M} = \frac{c(*,V,V)}{c(*,V)} = 0$$

$$q(NN|*,*) = \lambda_1 * \frac{c(*,*,NN)}{c(*,*)} + \lambda_2 * \frac{c(*,*)}{c(*)} \lambda_3 * \frac{c(*)}{M} = \frac{c(*,*,NN)}{c(*,*)} = \frac{1}{4}$$

$$q(V|*,NN) = \lambda_1 * \frac{c(*,NN,V)}{c(*,NN)} + \lambda_2 * \frac{c(*,NN)}{c(*)} \lambda_3 * \frac{c(NN)}{M} = \frac{c(*,NN,V)}{c(*,NN)} = 1$$

$$q(JJ|NN,V) = \lambda_1 * \frac{c(NN,V,JJ)}{c(NN,V)} + \lambda_2 * \frac{c(NN,V)}{c(V)} \lambda_3 * \frac{c(V)}{M} = \frac{c(NN,V,JJ)}{c(NN,V)} = \frac{1}{3}$$

E parameters:

$$e(Go|V) = \frac{c(Go,V)}{c(V)} = \frac{2}{3}$$

$$e(Go|NN) = \frac{c(Go,NN)}{c(NN)} = \frac{1}{2}$$

$$e(is|V) = \frac{c(is,V)}{c(V)} = \frac{1}{3}$$

$$e(great|JJ) = \frac{c(Go,V)}{c(V)} = \frac{1}{3}$$

Greedy algorithm would mistake in first tag in the sentence:

$$y_1 = \max_{y_1} e(Go|y_1) \cdot q(y_1|*,*) = V$$

For the second tag probability is zero for all options, because it has not been seen a word 'is' after a verb.

$$y_2 = \max_{y_2} e(is|y_2) \cdot q(y_2|*,V)$$

Therefore, the greedy algorithm would choose path with probability zero, even though the correct path probability is not zero, as can be seen with second order calculation:

$$p(Go, is, great, NN, V, JJ) = q(NN|*,*) \cdot q(V|*,NN) \cdot q(JJ|NN,V)$$
$$\cdot e(Go|NN) \cdot e(is|V) \cdot e(great|JJ) = \frac{1}{4} \cdot 1 \cdot \frac{1}{3} \cdot \frac{1}{2} \cdot \frac{1}{3} \cdot \frac{1}{3} > 0$$

(e) *keep in mind the ancient Chinese game 'Go' while reading this example.

Train sample:

x_1 =(He is a **great** man) $y_1 = (PRP, V, DT, JJ, NN)$

x_2 =(Go is a game) $y_2 = (NN, V, DT, NN)$

x_3 =(Go is **great**) $y_3 = (NN, V, JJ)$

x_4 =(My name is **great**) $y_4 = (PRP, NN, V, NN)$

Q parameters: (assume no smoothing for simplicity)

$$q(JJ|V,DT) = \frac{1}{2}$$

$$q(NN|V,DT) = \frac{1}{2}$$

$$q(NN|NN,V,DT) = 1$$

E parameters:

$$e(JJ|great) = \frac{c(great,JJ)}{c(JJ)} = \frac{2}{3}$$

$$e(NN|great) = \frac{c(great,NN)}{c(NN)} = \frac{1}{3}$$

Test Sentence s:

Go is a great game

(NN,V,DT,JJ,NN)

it's not hard to see the three first words would be tagged as (NN,V,DT) – since that's only possible path with non-zero probability as 'Go', 'is' and 'a' appear in training only as NN, V and DT accordingly so their emission parameters are implying this tagging.

Notice that both second order and third order would act the same since only '*' are considered before.

The interesting part is at the word 'great'.

The second order is considering:

$$P(JJ|great) = q(JJ|V,DT) \cdot e(JJ|great) = \frac{1}{2} \cdot \frac{2}{3} = \frac{1}{3}$$

Which is the correct tagging.

Second order would choose that because the only other possible tagging for the word great in NN and the probability is smaller:

$$P(NN|great) = q(NN|V,DT) \cdot e(NN|great) = \frac{1}{2} \cdot \frac{1}{3} = \frac{1}{6}$$

Bu third order would choose:

$$P(NN|great) = q(NN|NN, V, DT) \cdot e(NN|great) = 1 \cdot \frac{1}{3} = \frac{1}{3}$$

Because all other $q(tag|NN, V, DT)$ for $tag \neq NN$ is zero (since other tags have not appeared after this sequence).

Question 4

(a) In code
(b) In code
(c) Optimizations that were done: We used the same pruning policy from the previous question to tweak tags, resulting in a minimal set for those words. In addition, a cache dictionary was saved for (curr_word, next token, prev_word, prevprev_word, prev_ token, prevprev_token) for quickness of retrieval.

(d) Dev: Accuracy greedy memm : 0.956876137298
Dev: Accuracy Viterbi memm : 0.948301218935

(e) Results errors analysis:
As the greedy model is better (slightly) we shall analyze it.

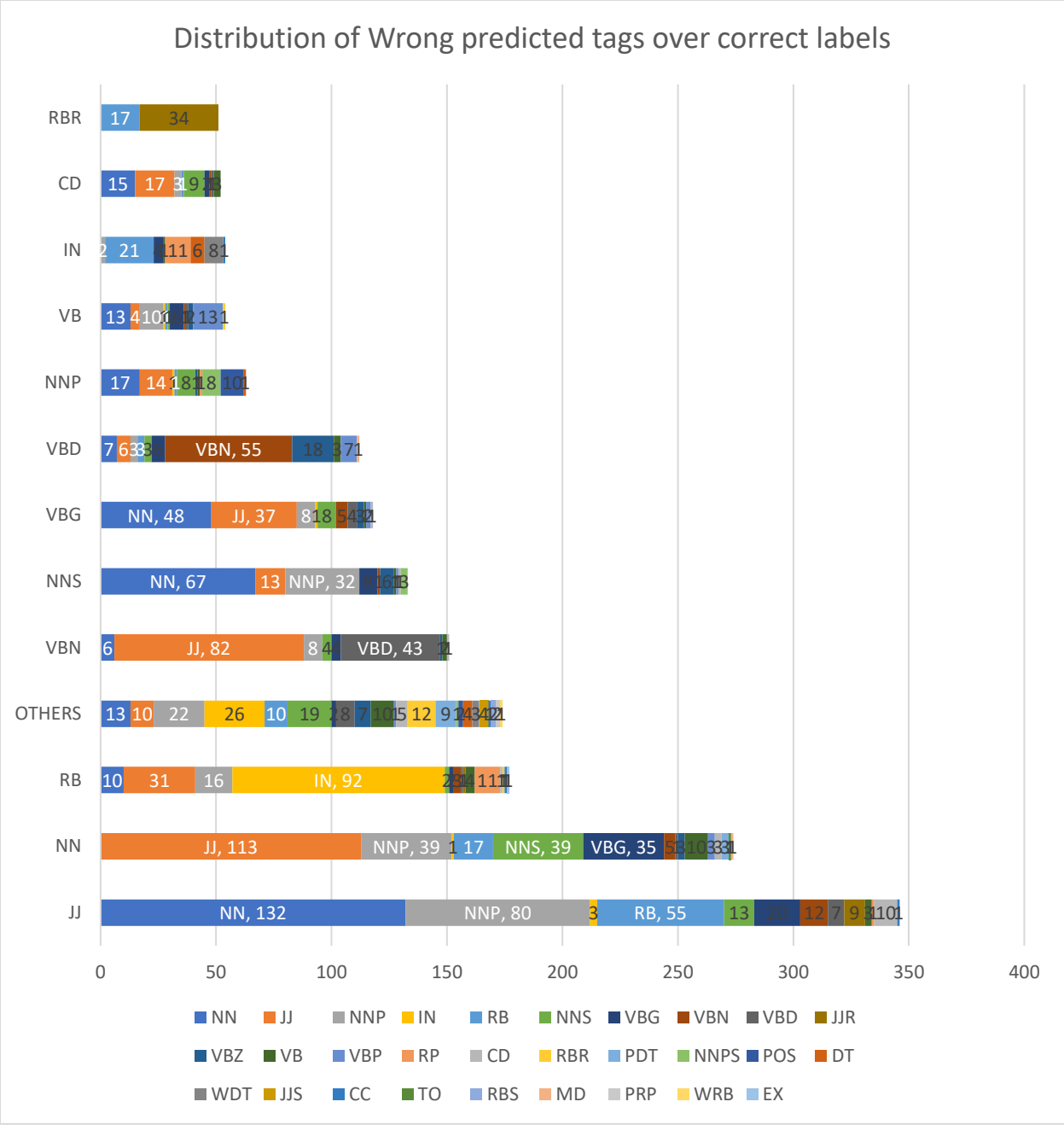
As can be seen the in chart below, the model tends to mix up the following:

If the initial tag was an Adjective, it tends to assign it Nouns or Adverbs. This makes sense, since many Adjectives can also be used as Nouns (almost any, by placing the adjective as the object of the sentence) and Adverbs (such as words that have to do with time, like ‘early’, ‘late’, etc.).

Nouns are also thought to be Adjectives more (for the same reasons), but Adverbs are less classified as Nouns mistakenly, and more as Pepositions (again, words that deal with time and directions, like ‘before’ or ‘up’, or words that describe the meaning like ‘about’ and ‘as’).

Nouns also tend to be confused with ing verbs (which again makes sense). And ing-verbs in turn are mixed up with Nouns and adjectives (e.g. of such words – ‘recycling’, ‘leading’).

Finally, past participle verbs also tend to be confused with adjectives (as many verbs in past tense can be used as adjective, like ‘reported’).


Distribution of Wrong predicted tags over correct labels
RBR
CD
IN
VB
NNP
VBD
VBG
NNS
VBN
OTHERS
RB
NN
JJ
VBN, 55
NN, 48
JJ, 37
NN, 67
NNP, 32
JJ, 82
VBD, 43
IN, 92
JJ, 113
NNP, 39
NNS, 39
VBG, 35
NN, 132
NNP, 80
RB, 55
0
50
100
150
200
250
300
350
400
NN
JJ
NNP
IN
RB
NNS
VBG
VBN
VBD
JJR
VBZ
VB
VBP
RP
CD
RBR
PDT
NNPS
POS
DT
WDT
JJS
CC
TO
RBS
MD
PRP
WRB
EX