

Yuan Tian
UNI:yt2583

Question4:

```
dyn-160-39-248-50:COMS4705 yt2583columbia.edu$ python2.7 count_freqs.py ner_train_new.dat > ner_new.counts
dyn-160-39-248-50:COMS4705 yt2583columbia.edu$ python2.7 eval_ne_tagger.py ner_dev.key 4_2.txt
Found 14043 NEs. Expected 5931 NEs; Correct: 3117.

      precision      recall      F1-Score
Total:  0.221961    0.525544    0.312106
PER:    0.435451    0.231230    0.302061
ORG:    0.475936    0.399103    0.434146
LOC:    0.147750    0.870229    0.252612
MISC:   0.491689    0.610206    0.544574
dyn-160-39-248-50:COMS4705 yt2583columbia.edu$
```

Figure 1

To run my code:

1. `python2.7 count_freqs.py ner_train.dat > ner.counts`
2. `python2.7 4_1.py replace words`
3. `python2.7 count_freqs.py ner_train_rare.dat > ner_rare.counts`
4. `python2.7 4_2.py generate 4_2.txt`
5. `python2.7 eval_ne_tagger.py ner_dev.key 4_2.txt observe the performance`

From the performance shown above, it can be seen that the baseline tagger has a low recall rate and precision. This is because this tagger does not take the previous context into consideration. F1 score is also very low indicating that this baseline model is not precise for tagging.

Question5:

1. `python2.7 5_1.py generate 5_1.txt`
2. `python2.7 5_2.py generates 5_2.txt`

1 O I-LOC I-MISC -5.67404646373167
2 O O B-MISC -11.148261551165996
3 I-ORG O I-MISC -5.161515325055391
4 I-MISC I-MISC I-MISC -1.3447659024233083
5 O I-ORG I-MISC -6.448889394146858
6 I-PER I-PER STOP -3.2418858873091896
7 * I-MISC STOP -6.218600119691729
8 * I-PER STOP -5.820824495087865
9 O I-PER I-PER -0.39462307338287467
10 O I-ORG STOP -4.081765780015241
11 I-LOC I-LOC O -0.16211192908515626
12 I-MISC I-MISC B-MISC -5.9331289700950185
13 O I-MISC I-MISC -1.4569375217819227
14 * I-ORG STOP -4.309374478762141
15 I-MISC I-LOC O -0.6931471805599453
16 O O I-LOC -3.9086877042439934
17 * I-ORG I-LOC -7.805882040228621
18 I-LOC I-LOC I-LOC -2.299130858125958
19 I-MISC I-MISC I-LOC -6.338594078203183
20 * O I-PER -2.1310317961026217
21 * * I-MISC -3.3311367804229133
22 O I-MISC B-MISC -4.7801733619743665
23 O I-MISC O -0.3287372759283168
24 I-PER I-PER O -0.09758750592646027
25 I-ORG I-MISC O -0.11778303565638351
26 I-LOC O STOP -1.4950822654240459
27 O I-LOC I-LOC -1.8886613724086343
28 I-ORG O I-LOC -3.327791656728304
29 I-LOC B-LOC I-LOC -2.3978952727983707
30 O B-MISC I-MISC 0.0
31 I-LOC I-LOC B-LOC -5.954108760564213
32 I-MISC I-PER I-PER -0.033336420267591836
33 B-MISC O I-ORG -1.540445040947149
34 I-LOC I-PER O 0.0
35 * I-MISC I-PER -4.272689970636416
36 I-PER O I-MISC -4.928198418340367

```

110 * I-LOC STOP -4.657762636107262
111 I-PER O I-ORG -3.515092799769819
112 I-ORG I-PER I-PER -0.1823215567939546
113 B-LOC O O 0.0
114 I-ORG I-ORG I-PER -7.11855630709796
115 I-PER O I-PER -2.46342053598298
116 I-ORG I-ORG I-MISC -7.11855630709796
117 * I-PER I-PER -0.6181923786742364
118 * I-MISC I-ORG -5.5254529391317835
119 B-MISC I-MISC O -0.13976194237515874
120 O I-PER STOP -4.442265454016395
121 * I-ORG I-ORG -1.022556839624661
122 * O STOP -4.169981976427721
123 I-ORG I-ORG O -0.43394457943003295
124 I-LOC O I-MISC -4.353101058868445
125 O I-MISC STOP -4.822732976393162
126 * I-LOC I-MISC -5.286371295529636
127 O I-PER I-MISC -8.553139318189707
128 B-ORG B-ORG B-ORG -0.40546510810816444
129 I-PER I-LOC I-LOC -1.0986122886681098
130 O I-MISC I-LOC -6.012317043266998
131 I-MISC O I-PER -3.1168670001622862
132 I-PER I-PER I-PER -2.920867394589817
133 I-LOC I-MISC I-MISC -1.3217558399823195
134 I-MISC B-MISC I-MISC -0.5108256237659907
135 I-ORG I-PER O -1.791759469228055
136 O O I-MISC -4.050299221115914
137 * O O -0.2853138544489416
138 I-ORG I-ORG STOP -3.9405024767500145
139 * O I-ORG -3.186180953026837
140 I-MISC O I-ORG -4.100097214711551
141 I-PER O STOP -2.7830302029602003
142 I-MISC O O -0.17242802099584573
143 I-LOC I-LOC I-ORG -7.052721049232323
144

```

This script calculates the log (base e). 0 shows the highest probability. This result meets with the files provided. Some of the results are verified by hands.

```

[dyn-160-39-248-50:COMS4705 yt2583columbia.edu$ python2.7 eval_ne_tagger.py ner_dev.key 5_2.txt
Found 4704 NEs. Expected 5931 NEs; Correct: 3648.

      precision      recall      F1-Score
Total:  0.775510      0.615073      0.686037
PER:    0.763231      0.596300      0.669517
ORG:    0.611855      0.478326      0.536913
LOC:    0.876458      0.696292      0.776056
MISC:   0.830065      0.689468      0.753262
dyn-160-39-248-50:COMS4705 yt2583columbia.edu$ █

```

Figure 2

As can be seen from the figure 2 above, after the HMM tagger is implemented, the tagger can achieve a better performance than the baseline tagger in question 4. The precision of all four types of tags increases, and the location tagging precision increases the most. Tagging location words increased a lot, while the recall rate of person grows rapidly. F1 score and recall rate rise as well. The total precision is around 77% and ORG recall rate scores lowest.

Question6:

1. python2.7 6_0.py FOR replacing words
2. python2.7 6.py

For this question, the HMM tagger is implemented after the “_RARE_” words are mapped into subclasses:

- (1) All digit class”_DIGIT_”: all characters are number
- (2) All upper class”_UPPER_”: all characters are uppercase
- (3) All lower class”_LOWER_”: all characters are lowercase
- (4) All numeral values class”NUMERALVALUES”: all characters are numbers or dash “-” or “,”
- (5) The rest are the “__RARE__” class: the words do not belong to the above class are mapped to rare

```
[dyn-160-39-248-50:COMS4705 yt2583columbia.edu$ python2.7 eval_ne_tagger.py ner_dev.key 6.txt
Found 5758 NEs. Expected 5931 NEs; Correct: 4285.

      precision      recall      F1-Score
Total:  0.744182      0.722475      0.733168
PER:    0.807037      0.773667      0.790000
ORG:    0.529059      0.659940      0.587296
LOC:    0.863898      0.737186      0.795528
MISC:   0.824147      0.681868      0.746286
dyn-160-39-248-50:COMS4705 yt2583columbia.edu$ █
```

As can be seen from the figure, the F-1 score and recall score above 70% , although the precision is a little bit lower. This shows mapping words to different groups can further improve the performance of the tagger.