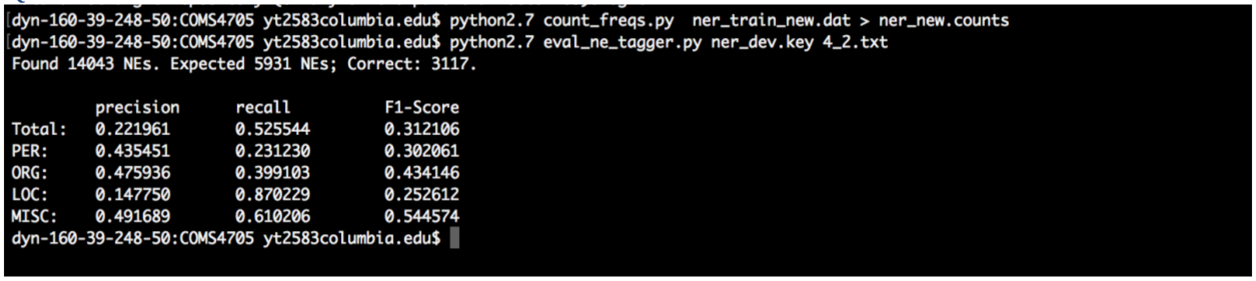
Yuan Tian
UNI:yt2583

My PC has version python3 and python2, so my command in terminal starts with python2.7. You can simply replace python2.7 with python in each command if your PC runs on python 2.

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:

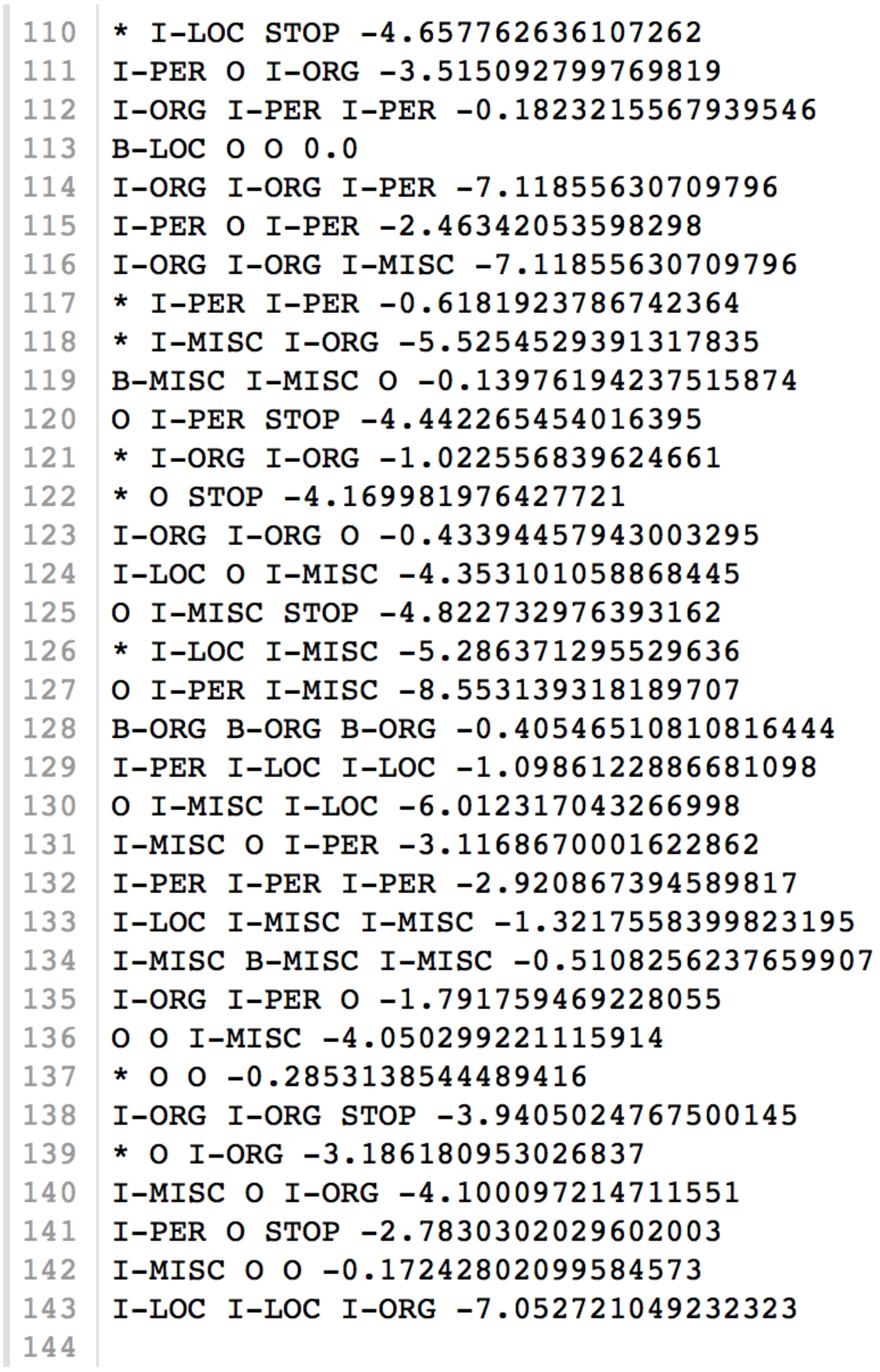
1. python2.7 count_freqs.py ner_train.dat > ner.counts
2. python2.7 4_1.py replace words
3. python2.7 count_freqs.py ner_train_rare.dat > ner_rare.counts
4. python2.7 4_2.py generate 4_2.txt
5. python2.7 eval_ne_tagger.py ner_dev.key 4_2.txt observe the performance

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:

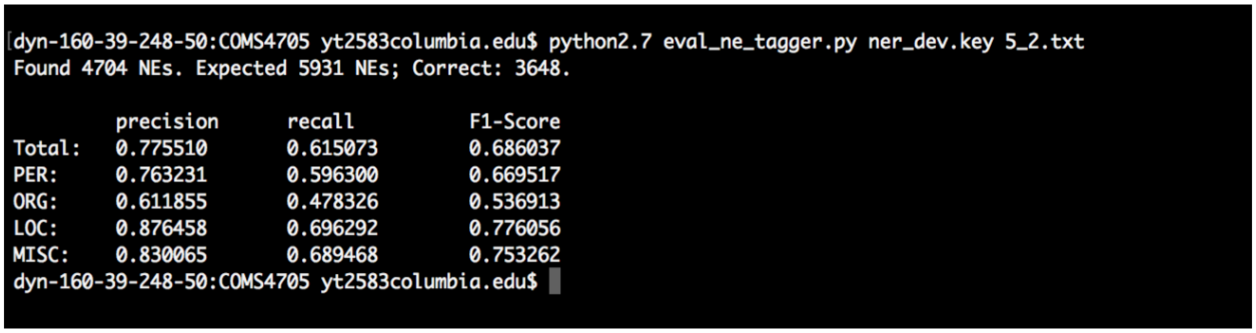
1. python2.7 5_1.py generate 5_1.txt
2. python2.7 5_2.py generates 5_2.txt
3. python2.7 eval_ne_tagger.py ner_dev.key 5_2.txt

```
O I-LOC I-MISC -5.67404646373167
O O B-MISC -11.148261551165996
I-ORG O I-MISC -5.161515325055391
I-MISC I-MISC I-MISC -1.3447659024233083
O I-ORG I-MISC -6.448889394146858
I-PER I-PER STOP -3.2418858873091896
* I-MISC STOP -6.218600119691729
* I-PER STOP -5.820824495087865
O I-PER I-PER -0.39462307338287467
O I-ORG STOP -4.081765780015241
I-LOC I-LOC O -0.16211192908515626
I-MISC I-MISC B-MISC -5.9331289700950185
O I-MISC I-MISC -1.4569375217819227
* I-ORG STOP -4.309374478762141
I-MISC I-LOC O -0.693147180559945З
O O I-LOC -3.9086877042439934
* I-ORG I-LOC -7.805882040228621
I-LOC I-LOC I-LOC -2.299130858125958
I-MISC I-MISC I-LOC -6.338594078203183
* O I-PER -2.1310317961026217
* * I-MISC -3.331136780422913З
O I-MISC B-MISC -4.7801733619743665
O I-MISC O -0.3287372759283168
I-PER I-PER O -0.09758750592646027
I-ORG I-MISC O -0.11778303565638351
I-LOC O STOP -1.4950822654240459
O I-LOC I-LOC -1.888661372408634З
I-ORG O I-LOC -3.327791656728304
I-LOC B-LOC I-LOC -2.3978952727983707
O B-MISC I-MISC 0.0
I-LOC I-LOC B-LOC -5.95410876056421З
I-MISC I-PER I-PER -0.033336420267591836
B-MISC O I-ORG -1.540445040947149
I-LOC I-PER O 0.0
* I-MISC I-PER -4.272689970636416
I-PER O I-MISC -4.928198418340367
```



```
* I-LOC STOP -4.657762636107262
I-PER O I-ORG -3.515092799769819
I-ORG I-PER I-PER -0.1823215567939546
B-LOC O O 0.0
I-ORG I-ORG I-PER -7.11855630709796
I-PER O I-PER -2.46342053598298
I-ORG I-ORG I-MISC -7.11855630709796
* I-PER I-PER -0.6181923786742364
* I-MISC I-ORG -5.5254529391317835
B-MISC I-MISC O -0.13976194237515874
O I-PER STOP -4.442265454016395
* I-ORG I-ORG -1.022556839624661
* O STOP -4.169981976427721
I-ORG I-ORG O -0.43394457943003295
I-LOC O I-MISC -4.353101058868445
O I-MISC STOP -4.822732976393162
* I-LOC I-MISC -5.286371295529636
O I-PER I-MISC -8.553139318189707
B-ORG B-ORG B-ORG -0.40546510810816444
I-PER I-LOC I-LOC -1.0986122886681098
O I-MISC I-LOC -6.012317043266998
I-MISC O I-PER -3.1168670001622862
I-PER I-PER I-PER -2.920867394589817
I-LOC I-MISC I-MISC -1.3217558399823195
I-MISC B-MISC I-MISC -0.5108256237659907
I-ORG I-PER O -1.791759469228055
O O I-MISC -4.050299221115914
* O O -0.2853138544489416
I-ORG I-ORG STOP -3.9405024767500145
* O I-ORG -3.186180953026837
I-MISC O I-ORG -4.100097214711551
I-PER O STOP -2.7830302029602003
I-MISC O O -0.17242802099584573
I-LOC I-LOC I-ORG -7.052721049232323
```

This script calculates the log (base e). the base can be changed to base 2 using log2(). 0.0shows 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 that the baseline tagger in question 4. The precision of all four types of tags increase from around 0.2 to around 0.7, and the location tagging precision increase the most. While the precision of tagging location words increases a lot, the recall rate of person becomes much bigger. F1 score and recall rate rise as well. The total precision is around 77% and ORG recall rate scores lowest, and the ORG precision need to be improved.

Question6:

Simply run:

1. python2.7 6.py

if the environment variable gets wrong

then please run the files separately.

run:

1. python2.7 utilfor6.py FOR replacing words
2. python2.7 6.py

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

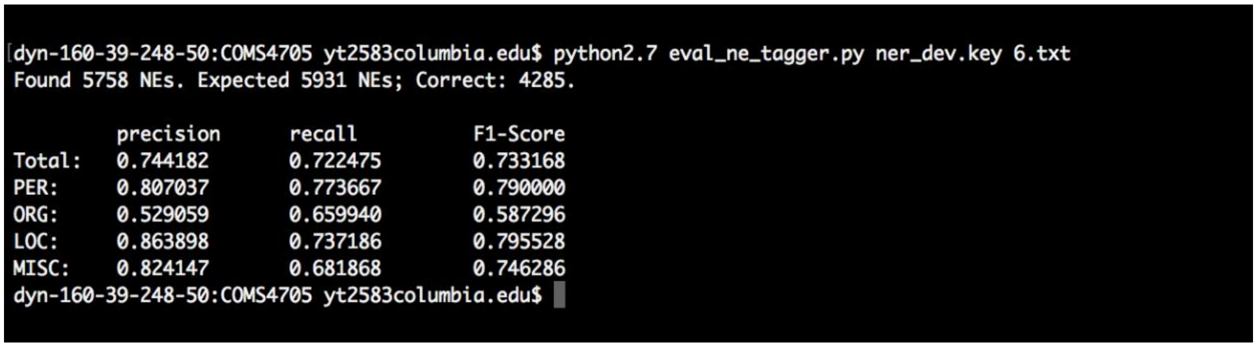
(1) All digit class"_DIGIT_": all characters are number

(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. The precision for ORG is the lowest, while the precision for MISC is the largest. This shows mapping words to different groups can further improve the performance of the tagger.