COMS 4705: Natural language Processing
Programming Assignment 1
@author: apoorv purwar (UNI - ap3644)

Code execution instructions –

- This code has been written in python3.
- For parts 4.1, 4.2, 5.1 and 5.2 the code generates all the files required to execute 'count' and 'evaluate' scripts in the format specified in the assignment. In order to test individual questions, you can execute individual files and then run the scripts to get count and performance metrics, in the order mentioned in the assignment.
  Also, please note that for your convenience all the evaluation results have also been added at the top of each script (where applicable) in comments.
- The only change you would need to do is in script 6.py -
  (Alternatively, use 'ner_rare_new.counts' and '6.txt' provided in the submission zip and just run evaluation script on 6.txt to get results)
  - If not using the file provided in the submission zip, you will have to create 'ner_rare_new.counts' and '6.txt'.
    To do so - open 6.py and comment line 329, which is a call to function 'writeViterbiTags' – the comment above this call also states the same.
  - Execute 6.py, this will generate 'ner_train_rare_new.dat'. Now run *"python count_freq.py ner_train_rare_new.dat > ner_rare_new.counts"* to generate 'ner_rare_new.counts' file. (Note that count_freq.py provided in the assignment is writte in python 2, so do not forget to switch environment)
  - Now, go back to 6.py and uncomment line 329, which you commented before and rerun 6.py. This would generate output file '6.txt' using the new counts which we generated in last step.
  - Now use the newly generated '6.txt' to evaluate the results.

For any real-time clarifications, I can be reached at – ap3644@columbia.edu / 571 665 1761

Results and Observations –

- For part 4_2: Trigram Tagger, the results obtained were as –

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

COMS 4705: Natural language Processing
Programming Assignment 1
@author: apoorv purwar (UNI - ap3644)

- For part 5 2: Viterbi Algorithm, the results obtained were as –

Found 4704 NEs. Expected 5931 NEs; Correct: 3643.

	precision	**recall**	**F1-Score**
Total:	0.774447	0.614230	0.685096
PER:	0.759749	0.593580	0.666463
ORG:	0.611855	0.478326	0.536913
LOC:	0.876458	0.696292	0.776056
MISC:	0.830065	0.689468	0.753262

- For part 6: Viterbi with Rare word classification, the results obtained were as –

Found 5822 NEs. Expected 5931 NEs; Correct: 4328.

	precision	**recall**	**F1-Score**
Total:	0.743387	0.729725	0.736493
PER:	0.805886	0.774755	0.790014
ORG:	0.541162	0.668161	0.597993
LOC:	0.840826	0.754635	0.795402
MISC:	0.826948	0.679696	0.746126

Observations -

As we can see that, the results from question 4 to 6 show a constant improvement in the number of NEs found and their accuracy.

Moreover, there is a considerable improvement in F1-Scores, as I move from trigram to Viterbi to Viterbi with rare word classifier.
This improvement is in line with my expectations, as with each step I have incorporated more and more information regarding the words in our algorithm, which helped in improving the predictions.

Also, in the third case, where I have used rare words classifier with Viterbi algorithm, I noticed that as I increase the number of ways in which I classify the rare words there is a constant improvement in the NEs correctly classified, as these additional classification take into account more information about the types of word which appear in our training data and hence a constant improvement in accuracy.