Homework 1

Name: Xinyue Tan

UNI: xt2215

You can running the whole stuff by running main.py, which will call each .py file to generate corresponding counting file and .txt file.

“ python main.py ”

Question 4

- Evaluation

The evaluation result is :

```
(Pycharm) dyn-160-39-172-69:hw1_xt2215 Seven$ python3 eval_ne_tagger.py ner_dev.key 4_2.txt
Found 13766 NEs. Expected 5931 NEs; Correct: 3144.

         precision      recall         F1-Score
Total:   0.228389       0.530096       0.319236
PER:     0.429461       0.225245       0.295583
ORG:     0.522908       0.392377       0.448335
LOC:     0.147927       0.877317       0.253157
MISC:    0.647123       0.647123       0.647123
```

- Observation

1. The precision and recall sometimes have huge difference. Mainly because we found much more name entities than the gold standard. 2. As we can see, for MISC the precision is equal to recall, which means the algorithm did not tag any word in the training data which is not MISC with MISC.

3. When it comes to LOC, things are getting more confusing. This one did pretty well on LOC recall but poor on precision. It has tagged lots of word with LOC by mistake (The precision is really low in LOC). I later checked the number of LOC appears in our training data, which is 8297. It is not the least entity (with MISC only 4593). Also, the log probability of all the words that tagged with B-LOC is 0.

In my point of view, as this algorithm did not take the context into consideration, only the name-entity and the entity itself have influence on the result. Thus, for a word which has once been tagged together with an entity that seldom appear in the training data, it is of high possibility we again tag this word with the entity that seldom appear. Because even this word has been tagged with a popular entity for much more times, the probability will be diluted by the denominator count(y) as y is very large.

I guess for the training data, a word is seldom marked with MISC and LOC at the same time, but it is probably marked with LOC and some other types. For example, the word Apple may be marked as LOC and ORG, but seldom marked as LOC and MISC together.

Question 5

- Evaluation

The evaluation result is:

```
(Pycharm) tanxiaoqideMacBook-Pro:hw1_xt2215 Seven$ python eval_ne_tagger.py ner_dev.key 5_2.txt
Found 4712 NEs. Expected 5931 NEs; Correct: 3577.

         precision      recall         F1-Score
Total:   0.759126       0.603102       0.672179
PER:     0.767921       0.588683       0.666461
ORG:     0.569597       0.464873       0.511934
LOC:     0.868239       0.682661       0.764347
MISC:    0.807542       0.674267       0.734911
```

- Observation

As is shown in the screen capture, when using Viterbi algorithm to predict the tag for test data, we found less NEs(5494), which is less than what we see in the gold standard file(5931).

I guess this is partially a result of simply defining infrequent words into _RARE_. As for any word appears in the test data while not being seen in the train data, it shares the same prediction with _RARE_. This may leads to less new combinations of name and entities.

Question 6

- Rules

Rule	Substitution
words presents weekdays, MONDAY - SATURDAY	_WEEKDAY_
words presents months, January - December	_MONTH_
words consists all capital letters	_UPPER_
words with the first letter being capital	_TITLE_
words consists of letters	_ALPHA_
words consists of numbers	_DIGIT_
words with special punctuation (including , . / etc.)	_SPECIAL_
Otherwise	_RARE_

- Evaluation

The evaluation result is:

```
(Pycharm) tanxiaoqideMacBook-Pro:hw1_xt2215 Seven$ python eval_ne_tagger.py ner_dev.key 6.txt
Found 5846 NEs. Expected 5931 NEs; Correct: 4277.

         precision      recall          F1-Score
Total:   0.731611       0.721126        0.726331
PER:     0.811245       0.769314        0.789724
ORG:     0.524179       0.656203        0.582808
LOC:     0.825694       0.746456        0.784078
MISC:    0.800000       0.668838        0.728563
```

- Observation

Both the precision and recall have been improved after specifying detailed rules for infrequent word. Also, the number of NEs we found is closer to the gold standard this time, resulting to similar precision and recall.

As I divided the infrequent word into eight groups, it has more possibility to be tagged with different entities. As the result shows, this will improve the performance of this algorithm.