Problem Set 1 Observations

UNI: yz3687 Name: Yonghe Zhao

All python files, including 4_1.py, 4_2.py, 5_1.py, 5_2.py, 6.py, can be executed by **"*python3 ./<name>.py*"**, or **"*chmod +x ./<name>.py*"** and then **"*./<name>.py*"**.

1. Question 4 Part 1

- Performance

The running time if we already have the dependency ner.counts:

```
w4118@w4118:~/Desktop/nlp$ ./4_1.py
Generating ./ner_train_rare.dat...
Running time: 0.31777119636535645 seconds.
```

The running time without the dependency ner.counts:

```
w4118@w4118:~/Desktop/nlp$ ./4_1.py
Generating ./ner.counts...
Generating ./ner_train_rare.dat...
Running time: 1.402613878250122 seconds.
```

2. Question 4 Part 2

- Performance

The running time if we already have the dependencies ner.counts, ner_train_rare.dat, ner_rare.counts:

```
w4118@w4118:~/Desktop/nlp$ ./4_2.py
Generating ./4_2.txt...
Running time: 0.20888590812683105 seconds.
```

The running time without the dependencies ner.counts, ner_train_rare.dat, ner_rare.counts:

```
w4118@w4118:~/Desktop/nlp$ ./4_2.py
Generating ./ner.counts...
Generating ./ner_train_rare.dat...
Generating ./ner_rare.counts...
Generating ./4_2.txt...
Running time: 2.221982955932617 seconds.
```

The performance:

```
w4118@w4118:~/Desktop/nlp$ python3 eval_ne_tagger3.py ner_dev.key 4_2.txt
Found 11636 NEs. Expected 5931 NEs; Correct: 3896.

         precision      recall         F1-Score
Total:   0.334823       0.656888       0.443559
PER:     0.192650       0.824266       0.312307
ORG:     0.475936       0.399103       0.434146
LOC:     0.807668       0.689204       0.743748
MISC:    0.537327       0.633008       0.581256
```

- Observation

The F-score of this tagging is significantly higher than the expected value, 30. It is because, in this implementation, I regard the I-TYPE and B-TYPE as one type. If I regard them as different type, which is implemented in 4_2_diff.py, the performance will be lower:

```
w4118@w4118:~/Desktop/nlp$ python3 eval_ne_tagger3.py ner_dev.key 4_2_diff.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
```

This is because the number of B-TYPE is much lower than that of I-TYPE and the occurrences of the B-TYPE depends highly on the context. Thus, without enough sample and context information, we cannot get a good tagging for B-TYPE. For the simplicity, the following implementations all regard the I-TYPE and the B-TYPE as different types.

Tagging using only emission parameter will also tag same words and all rare/unseen words with the same tag.

3. Question 5 Part 1

- Performance

The running time if we already have the dependencies ner.counts, ner_train_rare.dat, ner_rare.counts:

```
w4118@w4118:~/Desktop/nlp$ ./5_1.py
Generating ./5_1.txt...
Running time: 0.009197473526000977 seconds.
```

The running time without the dependencies ner.counts, ner_train_rare.dat, ner_rare.counts:

```
w4118@w4118:~/Desktop/nlp$ ./5_1.py
Generating ./ner.counts...
Generating ./ner_train_rare.dat...
Generating ./ner_rare.counts...
Generating ./5_1.txt...
Running time: 1.7384977340698242 seconds.
```

4. Question 5 Part 2

- Performance

The running time if we already have the dependencies ner.counts, ner_train_rare.dat, ner_rare.counts:

```
w4118@w4118:~/Desktop/nlp$ ./5_2.py
Generating ./5_2.txt...
Running time: 2.015223264694214 seconds.
```

The running time without the dependencies ner.counts, ner_train_rare.dat, ner_rare.counts:

```
w4118@w4118:~/Desktop/nlp$ ./5_2.py
Generating ./ner.counts...
Generating ./ner_train_rare.dat...
Generating ./ner_rare.counts...
Generating ./5_2.txt...
Running time: 2.9814844131469727 seconds.
```

The performance:

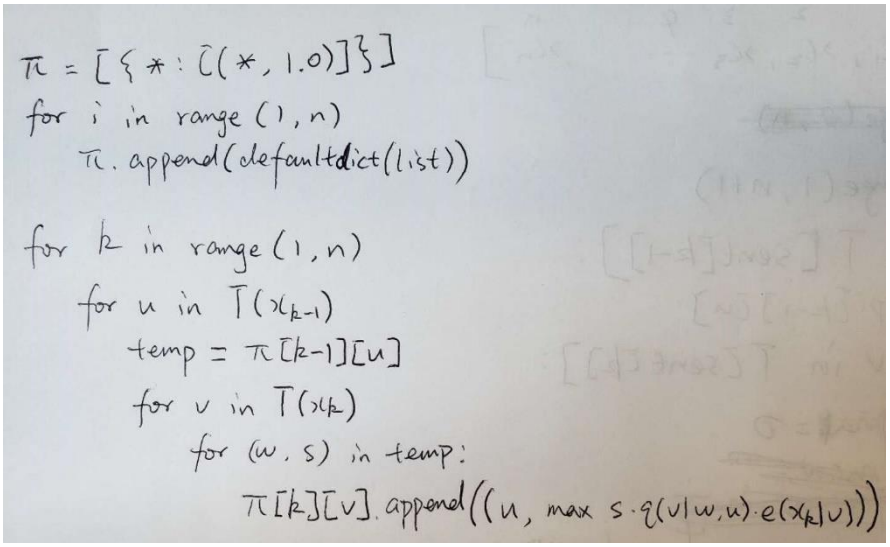
```
w4118@w4118:~/Desktop/nlp$ python3 eval_ne_tagger3.py ner_dev.key 5_2.txt
Found 4576 NEs. Expected 5931 NEs; Correct: 3607.

         precision      recall         F1-Score
Total:   0.788243       0.608161       0.686590
PER:     0.751960       0.573993       0.651034
ORG:     0.669483       0.473842       0.554923
LOC:     0.889889       0.700654       0.784015
MISC:    0.809463       0.687296       0.743394
```

- Observation

We can find the precision, recall and F-score are better than that in 4_2, which is reasonable. This time, because we introduce the context information, same words and rare/unseen words can be tagged different tags.

To improve the running time, besides using $T(x)$, which indicates the tag dictionary that lists the tags y such that $e(x|y) > 0$, I also optimized the data structure to store $\pi(k,u,v)$. I did not use a simple dictionary, where key is the tuple (k,u,v) and value is score, because it will cause $O(n|T(x)|^3)$ times dictionary accesses, which costs more than ordinary operations. Even worse, letting (k,u,v) tuple as key will increase the size of the dictionary and hence increase the possibility of hash crashes and refactoring. Observing that k is an increasing integer, so we can, obviously, regard k as the index of a $\pi(u,v)$ list. Also, in the inner loop, we need to loop over all ws, but we can find that we do not need to know the w firstly. We just need to know the best w after the loop. Thus, I designed the pseudocode like this:



We can see we only need $O(n|T(x)|^2)$ times accesses towards some smaller dictionaries this time.

5. Question 6

- Performance

The running time if we already have the dependencies ner.counts, ner_train_multirare.dat, ner_multirare.counts:

```
w4118@w4118:~/Desktop/nlp$ ./6.py
Generating ./6.txt...
Running time: 1.2368519306182861 seconds.
```

The running time without the dependencies ner.counts, ner_train_multirare.dat, ner_multirare.counts:

```
w4118@w4118:~/Desktop/nlp$ ./6.py
Generating ./ner.counts...
Generating ./ner_train_multirare.dat...
Generating ./ner_multirare.counts...
Generating ./6.txt...
Running time: 3.82381272315979 seconds.
```

The performance:

```
w4118@w4118:~/Desktop/nlp$ python3 eval_ne_tagger3.py ner_dev.key 6.txt
Found 5901 NEs. Expected 5931 NEs; Correct: 4262.

         precision      recall         F1-Score
Total:   0.722250       0.718597       0.720419
PER:     0.711716       0.796518       0.751733
ORG:     0.541509       0.643498       0.588115
LOC:     0.880759       0.708833       0.785498
MISC:    0.818766       0.691640       0.749853
```

- Observation

Informative patterns that is useful under this train and test set:

Informative patterns	Example(s)	Improvement on Performance		
		Precision	Recall	F-score
Number dominated words	1990-02-01; 3,900; 1.5;	+0.0067	+0.0137	+0.0113
First capital but not Digit and alpha	Sydney; A234-21(×)	-0.0660	+0.1104	+0.0339
other rare words	hahaha	-	-	-

After observing the train and test set, I found a lot of sentence contains only “all capital” words and these words has no special semantics, which maybe the reason of the decrease of the precision when using “First capital but not Digit and alpha” as a pattern. Of course, most sentences are normal, so the recall and the F-score increase.

Also, in this tagging task, the “Number dominated words” always mean nothing, so using that pattern can improve the performance.

Informative patterns that is tried but not useful under this training and test set:

Informative patterns	Example(s)
Two digits number	90
Four digits number	1990
All capitals	BBC

All digits	43258
Digit and alpha	A234-21
Capital and period	M.