CENG463 - Assignment 2 Report

Ali Dogan

January 16th 2021

1 Introduction

I have implemented all models stated in the homework. However, I have implemented them in a different structure from the one given as template. In the implementation part, I will explain this in detail.
Then, I will briefly explain my feature selection process on each tasks. After that, I will talk about each task in detail. I will discuss the performance metrics of the models under their section, with confusion matrices as well. I will also explain my feature selection process in more detail about each task. I will conclude all at the end, under the conclusions chapter.

2 Implementation

I have implemented two .py files named *tagger_lr.py* and *tagger_crf.py*. One for all Maximum Entropy Classifiers models and another for Conditional Random Fields models. Each file trains and tests the 3 tasks in order as provided so that I can use, for example, the predicted pos tags to use in the chunking model. When the files are run in the same folder with 'data' folder, it will automatically train the 3 tagger tasks and print the performance matrices.

3 Feature Selection

For all the tasks, I started with only using the word itself as a feature. Then I tested with adding some properties such as the uppercase of lowercase existence, or the prefix. Then I used the features used in sklearn-crfsuite tutorial and ended up with the following: Although they differ in some parts, I generally use the same information in all tasks. The predicted pos

```python
def feature_extraction(sent,i):
    word = sent[i][0]
    pos = sent[i][1]
    chunk = sent[i][2]
    features = {
        'bias' : 1.0,
        'word.lower()': word.lower(),
        'word[-3:]': word[-3:],
        'word[:2]': word[:2],
        'word[:3]': word[:3],
        'word.isupper()': word.isupper(),
        'word.istitle()': word.istitle(),
        'word.isdigit()': word.isdigit(),
        'pos':pos,
        'chunk':chunk
        }

    if i > 0:
        word1 = sent[i-1][0]
        features.update({
            '-1:word.lower()': word1.lower(),
            '-1:word.istitle()': word1.istitle(),
            '-1:word.isupper()': word1.isupper(),
            'pos-1': sent[i-1][1]
        })
    else:
        features['BOS'] = True

    if i < len(sent)-1:
        word1 = sent[i+1][0]
        features.update({
            '+1:word.lower()': word1.lower(),
            '+1:word.istitle()': word1.istitle(),
            '+1:word.isupper()': word1.isupper(),
            'pos+1': sent[i-1][1]
        })
    else:
        features['EOS'] = True

    return features
```

Figure 1: Feature Selection, adapted from sklearn documentation

and chunking information were different and the information about the neighbour words were useful in the maxent classifiers for all tasks. However, they were not contributing a lot in the CRF models. This is probably because of the fact that CRF already uses the context of the words.

4 Hyper-parameter Search

In the parameter search, I did not spend a lot of time. In the MaxEnt classifiers, I focused on the iteration number, and on the CRF I did optimization on c1 and c2 parameters as well as the iteration number. I did not find significantly different results.

5 Part of Speech Tagging

5.1 Maximum Entropy Classifiers - Multinomial Logistic Regression

Test Accuracy : 0.85
Unknown words Accuracy : 0.55
These accuracies are with the features about neighbour. Without changing anything else bu excluding neighbour information (context), I have received a slightly less accuracy:
Test Accuracy : 0.82
Unknown words Accuracy : 0.51
I have also tried training with using upos as a feature as well. However, I did not train it myself on the test set phase, rather used already existed labels. I tested just to compare and measure the benefit of having upos tags.
Test Accuracy : 0.91
Unknown words Accuracy : 0.74
Although the test accuracy increased slightly, a dramatic increase occurred in the accuracy of unknown words tagging. Having a prior knowledge about the upos tag of the unknown word contributed to the accuracy more than it contributed to encountered words in the training process. I believe, predicting the upos tags at the test time and use them in the prediction process of pos tags will show the similar results. Moreover, I believe adding the predicted upos tags of the neighbour words will also increase the accuracy significantly.
Below are Most informative features and Classification reports of the pos tagging maxent classifier that uses the features with neighbour information but without upos (first accuracy results above). I did not put the Confusion matrix for pos tagging because it is a big matrix in size (49x49) which makes it hard to read.
We see that upos features contributes to the model a lot. I believe that having a predicted upos tags from another model will have the same effect.

```
In [20]: print(cr_pos_test)
              precision    recall  f1-score   support

           $       1.00      0.93      0.96        14
          ''       0.88      0.77      0.82        88
           ,       0.92      0.98      0.95       936
       -LRB-       0.98      0.98      0.98       117
       -RRB-       0.97      0.99      0.98       120
           .       0.96      0.99      0.97      1503
           :       1.00      0.89      0.94       106
         ADD       1.00      0.40      0.57        80
         AFX       0.00      0.00      0.00         4
          CC       0.99      0.98      0.99       781
          CD       0.99      0.90      0.94       378
          DT       0.94      0.98      0.96      1943
          EX       1.00      0.80      0.89        56
          FW       1.00      0.30      0.46        30
          GW       0.33      0.03      0.06        32
        HYPH       0.72      0.41      0.52        95
          IN       0.90      0.99      0.94      2353
          JJ       0.93      0.95      0.94      1656
         JJR       1.00      0.68      0.81        47
         JJS       0.97      0.70      0.81        84
          LS       1.00      0.80      0.89         5
          MD       1.00      0.95      0.97       358
         NFP       0.92      0.60      0.73        60
          NN       0.80      0.98      0.88      3342
         NNP       0.91      0.96      0.94      1817
        NNPS       1.00      0.21      0.35        62
         NNS       0.93      0.66      0.77       930
         PDT       0.67      0.19      0.30        21
         POS       0.99      0.85      0.91        84
         PRP       0.98      0.98      0.98      1487
        PRP$       0.98      0.90      0.94       315
          RB       0.95      0.87      0.91      1292
         RBR       0.77      0.45      0.57        22
         RBS       1.00      0.55      0.71        20
          RP       0.92      0.63      0.75        76
         SYM       1.00      0.18      0.30        17
          TO       0.99      0.94      0.96       359
          UH       1.00      0.54      0.70       116
          VB       0.81      0.84      0.83      1122
         VBD       0.89      0.74      0.81       520
         VBG       0.98      0.77      0.86       384
         VBN       0.85      0.76      0.80       476
         VBP       0.92      0.74      0.82       771
         VBZ       0.98      0.91      0.94       643
         WDT       0.88      0.46      0.60       106
          WP       0.93      0.96      0.94       113
         WP$       0.00      0.00      0.00         2
         WRB       1.00      0.99      1.00       113
          XX       0.00      0.00      0.00         3
          ``       0.81      0.91      0.86        91

    accuracy                           0.91     25150
   macro avg       0.87      0.70      0.75     25150
weighted avg       0.91      0.91      0.91     25150
```

Figure 2: Classification Report of POS tagging with MaxEnt

```
In [36]: clf_pos.show_most_informative_features(15)
  -4.329 upos=='NOUN' and label is 'NNP'
  -4.312 upos=='PROPN' and label is 'NN'
  -4.228 upos=='NOUN' and label is 'JJ'
  -4.059 EOS==True and label is 'VBP'
  -3.856 upos=='ADJ' and label is 'NN'
  -3.820 EOS==True and label is 'VBG'
  -3.796 EOS==True and label is 'MD'
  -3.766 word.istitle()==True and label is 'RP'
  -3.664 upos=='NOUN' and label is 'VBG'
  -3.624 word[:2]=='th' and label is 'VBZ'
  -3.456 upos=='PROPN' and label is 'JJ'
  -3.453 BOS==True and label is '.'
  -3.447 word[:2]=='th' and label is 'NNP'
  -3.375 EOS==True and label is '``'
  -3.361 EOS==True and label is 'VBZ'
```

Figure 3: Most informative features of POS tagging with MaxEnt

5.2 Conditional Random Fields

In the CRF model, I achieved very good results in all tasks. The importance of upos tags was significant but in this part, I will show the results obtained without using any upos,pos or chunk features in any part, However I will share the accuracy of both. I also realized that less features were needed compared to maxent models.
Test Accuracy with upos : 0.98
Test Accuracy without upos : 0.93

	precision	recall	f1-score	support
$	1.00	0.93	0.96	14
''	0.89	0.88	0.88	88
,	0.96	0.98	0.97	936
-LRB-	0.98	1.00	0.99	117
-RRB-	0.97	1.00	0.98	120
.	1.00	0.99	1.00	1503
:	0.91	0.91	0.91	106
ADD	0.97	0.88	0.92	80
AFX	0.00	0.00	0.00	4
CC	0.99	0.99	0.99	781
CD	0.98	0.94	0.96	378
DT	0.98	0.99	0.99	1943
EX	0.96	0.91	0.94	56
FW	1.00	0.27	0.42	30
GW	0.09	0.09	0.09	32
HYPH	0.88	0.80	0.84	95
IN	0.96	0.97	0.96	2353
JJ	0.90	0.87	0.88	1656
JJR	0.80	0.77	0.78	47
JJS	0.90	0.90	0.90	84
LS	1.00	0.80	0.89	5
MD	1.00	0.99	0.99	358
NFP	0.88	0.83	0.85	60
NN	0.88	0.92	0.90	3342
NNP	0.87	0.86	0.87	1817
NNPS	0.94	0.47	0.62	62
NNS	0.94	0.97	0.95	930
PDT	0.91	1.00	0.95	21
POS	0.92	0.99	0.95	84
PRP	0.99	0.99	0.99	1487
PRP$	1.00	0.99	1.00	315
RB	0.91	0.89	0.90	1292
RBR	0.47	0.41	0.44	22
RBS	0.93	0.70	0.80	20
RP	0.69	0.57	0.62	76
SYM	0.75	0.35	0.48	17
TO	0.97	0.98	0.97	359
UH	0.87	0.68	0.76	116
VB	0.89	0.89	0.89	1122
VBD	0.94	0.89	0.91	520
VBG	0.89	0.96	0.92	384
VBN	0.82	0.88	0.85	476
VBP	0.91	0.90	0.90	771
VBZ	0.98	0.96	0.97	643
WDT	0.85	0.87	0.86	106
WP	0.97	0.93	0.95	113
WP$	0.00	0.00	0.00	2
WRB	1.00	0.98	0.99	113
XX	0.00	0.00	0.00	3
``	0.87	0.90	0.89	91
accuracy			0.93	25150
macro avg	0.84	0.79	0.81	25150
weighted avg	0.93	0.93	0.93	25150

Figure 4: Classification Report of POS tagging with CRF

6 Chunking

In the chunking part, I followed a similar path as in POS tag. Different from that, I have added the pos feature to my feature set and experiment its contribution to the results.

6.1 Maximum Entropy Classifiers - Multinomial Logistic Regression

Test accuracy with using ground truth pos tags: 0.88
Test accuracy with using predicted pos tags: 0.84
Test accuracy without using any pos tag as feature: 0.85
As seen above, the pos tags did not give a lot of impact on accuracy even though stated otherwise in the assignment documentation. I think the reason is due to the low accuracy I got overall. I believe that to get higher accuracy, having pos tags is essential and it will show its impact on the further iterations of the training process. Below figures are from the model with using ground truth pos tags. I chose to use this to get a model that benefits the most with the pos tags so that I can get more sense about its impact on the chunking task.

In [21]: print(cm_chunk_test_gt)

	B-ADJP	B-ADVP	B-CONJP	B-INTJ	B-LST	B-NP	B-PP	B-PRT	B-SBAR	B-VP	I-ADJP	I-ADVP	I-CONJP	I-INTJ	I-NP	I-PP	I-SBAR	I-VP	O
B-ADJP	<105>	23	.	.	.	108	11	.	.	24	.	.	.	.	383	.	.	44	6
B-ADVP	3	<833>	.	.	.	199	67	2	.	11	.	.	.	.	179	.	.	53	7
B-CONJP	.	.	<.>	.	.	1	6	.	.	.	.	.	.	.	2	.	.	3	6
B-INTJ	.	2	.	<.>	.	6	.	.	.	2	.	.	.	.	2	.	.	.	.
B-LST	.	.	.	.	<.>	.	.	.	.	.	.	.	.	.	1	.	.	.	.
B-NP	.	31	.	.	.	<15030>	48	2	5	53	.	.	.	.	2700	.	.	51	33
B-PP	4	13	.	.	.	41	<6560>	.	3	129	.	.	.	.	42	.	.	3	27
B-PRT	.	59	.	.	.	6	62	<29>	.	1	.	.	.	.	10	.	.	4	.
B-SBAR	.	13	.	.	.	242	254	.	<215>	.	.	.	.	.	5	.	.	.	1
B-VP	.	21	.	.	.	76	74	.	.	<6379>	.	.	.	.	288	.	.	219	5
I-ADJP	4	4	.	.	.	22	6	.	.	6	<.>	1	.	.	165	.	.	4	11
I-ADVP	1	20	.	.	.	23	11	.	.	1	.	<4>	.	.	81	.	.	5	14
I-CONJP	.	3	.	.	.	.	11	.	.	.	.	.	<.>	.	7	.	.	.	.
I-INTJ	.	.	.	.	.	.	.	.	.	1	.	.	.	<.>	.	.	.	.	.
I-NP	1	9	.	.	.	733	79	.	.	72	.	.	.	.	<19054>	.	.	28	629
I-PP	.	1	.	.	.	1	74	.	.	1	.	.	.	.	.	<.>	.	.	8
I-SBAR	.	.	.	.	.	3	7	.	14	.	.	.	.	.	.	.	<.>	.	.
I-VP	.	52	.	.	.	47	40	.	.	465	.	1	.	.	205	.	.	<3144>	48
O	.	6	.	.	.	91	18	.	5	27	.	.	.	.	105	.	.	82	<8824>

(row = reference; col = test)

Figure 5: Confusion Matrix of Chunk tagging with MaxEnt

In [23]: print(cr_chunk_test_gt)

	precision	recall	f1-score	support
B-ADJP	0.89	0.15	0.26	704
B-ADVP	0.76	0.62	0.68	1354
B-CONJP	0.00	0.00	0.00	18
B-INTJ	0.00	0.00	0.00	12
B-LST	0.00	0.00	0.00	1
B-NP	0.90	0.84	0.87	17953
B-PP	0.90	0.96	0.93	6822
B-PRT	0.88	0.17	0.28	171
B-SBAR	0.89	0.29	0.44	730
B-VP	0.89	0.90	0.90	7062
I-ADJP	0.00	0.00	0.00	223
I-ADVP	0.67	0.03	0.05	160
I-CONJP	0.00	0.00	0.00	21
I-INTJ	0.00	0.00	0.00	1
I-NP	0.82	0.92	0.87	20605
I-PP	0.00	0.00	0.00	85
I-SBAR	0.00	0.00	0.00	24
I-VP	0.86	0.79	0.82	4002
O	0.92	0.96	0.94	9158
accuracy			0.87	69106
macro avg	0.49	0.35	0.37	69106
weighted avg	0.87	0.87	0.86	69106

Figure 6: Classification Report of Chunk tagging with MaxEnt

From figures, we have fairly a balance model. However, there are some chunks that are not even once predicted such as B-CONTJ,B-INTJ,B-LST, I-ADJP, I-CONJP, I-INTJ, I-PP. Some of them are very rare in the dataset(X-INTJ have only 1 instance each), but some of them are not that rare(I-ADJP,I-PP,I-ADVP). This is one of the benefits of confusion matrices. It suggests further trials on the feature selection process. Because it seems that our model is not able to catch any information about these tags. This further research may involve looking at the data and these tags behaviour, i.e., their properties in English literature. However, I did not go into more detail on the analysis of this problem.

```
In [37]: clf_chunk.show_most_informative_features(15)
  -3.892 EOS==True and label is 'B-VP'
  -3.418 EOS==True and label is 'B-PP'
  -3.400 EOS==True and label is 'B-ADJP'
  -3.240 pos=='IN' and label is 'B-VP'
   3.137 word.lower()=='personal-injury' and label is 'B-UCP'
  -3.059 EOS==True and label is 'B-ADVP'
  -3.045 word.istitle()==True and label is 'I-ADVP'
  -3.043 pos=='CD' and label is 'B-VP'
   3.025 word.lower()=='wine' and label is 'B-UCP'
   2.954 -1:word.lower()=='wine' and label is 'I-UCP'
  -2.953 word.istitle()==True and label is 'I-PP'
   2.939 -1:word.lower()=='personal-injury' and label is 'I-UCP'
   2.915 +1:word.lower()=='soft-drink' and label is 'I-UCP'
   2.908 +1:word.lower()=='spirits' and label is 'I-UCP'
   2.771 +1:word.lower()=='lawyers' and label is 'I-UCP'
```

Figure 7: Classification Report of Chunk tagging with MaxEnt

Looking at the most informative features, a balance seems to occur. There are features from the context, the property of the word as well as its tag.

6.2 Conditional Random Fields

```
In [8]: print(cm_chunk)
```

	B-ADJP	B-ADVP	B-CONJP	B-INTJ	B-NP	B-PP	B-PRT	B-SBAR	B-VP	I-ADJP	I-ADVP	I-CONJP	I-INTJ	I-NP	I-PP	I-SBAR	I-VP	O
B-ADJP	<335>	14	.	.	28	1	3	.	10	10	2	.	.	28	.	.	30	6
B-ADVP	21	<795>	3	1	43	24	19	3	17	3	3	1	.	20	1	.	2	3
B-CONJP	.	.	<5>	.	.	1	.	.	.	.	.	.	.	2	.	.	.	1
B-INTJ	.	.	.	<1>	4	.	.	.	1	.	.	.	.	.	.	.	.	.
B-NP	17	31	.	.	<11381>	7	.	26	65	1	6	.	.	348	.	.	40	31
B-PP	3	21	3	.	12	<4465>	10	14	26	1	1	.	.	14	2	.	4	27
B-PRT	.	13	.	.	2	18	<83>	.	.	.	.	.	.	1	.	.	.	.
B-SBAR	1	5	1	.	18	48	.	<364>	.	.	.	1	.	4	.	.	.	2
B-VP	6	22	.	.	63	9	.	.	<4329>	3	1	2	.	135	.	.	85	3
I-ADJP	6	4	.	.	2	.	.	1	4	<95>	4	.	.	27	.	.	4	5
I-ADVP	3	4	.	.	7	4	.	.	3	6	<50>	.	.	18	2	.	.	1
I-CONJP	.	.	.	.	.	2	.	.	.	.	.	<8>	.	2	.	.	.	.
I-INTJ	.	.	.	.	.	.	.	.	2	.	.	.	<.>	.	.	.	.	.
I-NP	10	8	.	.	284	25	.	2	72	12	9	.	.	<13011>	.	.	7	141
I-PP	.	.	1	.	1	17	.	1	.	.	1	6	.	1	<32>	.	.	5
I-SBAR	.	.	.	.	.	.	.	3	.	.	.	.	.	1	.	<12>	.	.
I-VP	7	7	.	.	26	8	.	.	39	4	1	.	.	9	.	.	<2429>	10
O	1	11	1	.	47	8	.	13	12	2	4	1	.	148	.	.	21	<5954>

(row = reference; col = test)

Figure 8: Confusion Matrix of Chunk tagging with CRF

In the confusion matrix, we see the problem we discussed in the maxent model that some tags are not predicted, is now gone significantly. We can say that the CRF captures the property of the tags from the context much better that we tried with feature selection.

	precision	recall	f1-score	support
B-ADJP	0.82	0.72	0.76	467
B-ADVP	0.85	0.83	0.84	959
B-CONJP	0.36	0.56	0.43	9
B-INTJ	0.50	0.17	0.25	6
B-NP	0.95	0.95	0.95	11953
B-PP	0.96	0.97	0.97	4603
B-PRT	0.72	0.71	0.72	117
B-SBAR	0.85	0.82	0.84	444
B-VP	0.95	0.93	0.94	4658
I-ADJP	0.69	0.62	0.66	152
I-ADVP	0.61	0.51	0.56	98
I-CONJP	0.42	0.67	0.52	12
I-INTJ	0.00	0.00	0.00	2
I-NP	0.94	0.96	0.95	13581
I-PP	0.86	0.49	0.63	65
I-SBAR	1.00	0.75	0.86	16
I-VP	0.93	0.96	0.94	2540
O	0.96	0.96	0.96	6223
accuracy			0.94	45905
macro avg	0.74	0.70	0.71	45905
weighted avg	0.94	0.94	0.94	45905

Figure 9: Classification Report of Chunk tagging with CRF

7 Named Entity Recognition

I achieved good accuracy in this part, however, did not realize the imbalanced data problem at first. As it was stated, the 'O' tags are majority class in the dataset. So, detailed look on the confusion matrices and performance metrics are important.

7.1 Maximum Entropy Classifiers - Multinomial Logistic Regression

Test accuracy with using ground truth pos and chunking tags/without 'O': 0.954/0.78
Test accuracy with using predicted pos and chunking tags: 0.946
Test accuracy without using any pos and chunking tags: 0.947
In the named entity recognition part, I received almost equal results with or without pos/chunk features. I think adding the pos and chunk feature about the neighbour words would be make the difference. However, I did not train such model due to time limits and considered that we will see the difference when we train CRF since it will use the context.

```
In [22]: print(cm_ner_val_gt)
       |          B                   I                  |
       |    B     -     B     B     I     -     I     I        |
       |    -     M     -     -     -     M     -     -        |
       |    L     I     O     P     L     I     O     P        |
       |    O     S     R     E     O     S     R     E        |
       |    C     C     G     R     C     C     G     R     O  |
-------+-------------------------------------------------------+
 B-LOC | <1496>   13    67    29     1     .     1     3   227 |
B-MISC |    39 <629>    24    23     1     3     4     5   194 |
 B-ORG |    64    20 <833>    69     .     .    17     3   335 |
 B-PER |    50     5    42 <1460>    .     .     1    25   259 |
 I-LOC |     7     .     .     3 <169>     1    20    22    35 |
I-MISC |     7    18     6     3     7 <175>    12    11   107 |
 I-ORG |    25     7     7     5    12     6 <410>    51   228 |
 I-PER |     3     .     1    12     2     .    12 <1173>  104 |
     O |     9    12    23    15     .     3    16     3<42678>|
-------+-------------------------------------------------------+
(row = reference; col = test)
```

Figure 10: Confusion Matrix of NER tagging with MaxEnt

The existence of 'O' tag is making a lot of difference on the accuracy and weighted avg of the test.(0.95 vs 0.78) due to imbalance dataset. We should use the 0.78 accuracy as it is not exploiting the problems in the data distribution. Moreover, we see that the context is very important in the NER task. We will further see that CRF is more suitable in NER modeling.

```
In [24]: print(cr_ner_val_gt)
              precision    recall  f1-score   support

       B-LOC       0.88      0.81      0.85      1837
      B-MISC       0.89      0.68      0.77       922
       B-ORG       0.83      0.62      0.71      1341
       B-PER       0.90      0.79      0.84      1842
       I-LOC       0.88      0.66      0.75       257
      I-MISC       0.93      0.51      0.66       346
       I-ORG       0.83      0.55      0.66       751
       I-PER       0.91      0.90      0.90      1307
           O       0.97      1.00      0.98     42759

    accuracy                           0.95     51362
   macro avg       0.89      0.72      0.79     51362
weighted avg       0.95      0.95      0.95     51362
```

Figure 11: Classification Report of NER tagging with MaxEnt including ’O’

```
In [35]: print(a)
              precision    recall  f1-score   support

       I-PER       0.83      0.82      0.83      1307
      B-MISC       0.90      0.69      0.78       922
       I-LOC       0.83      0.67      0.74       257
       B-ORG       0.85      0.60      0.70      1341
       B-LOC       0.88      0.78      0.83      1837
       I-ORG       0.84      0.50      0.63       751
      I-MISC       0.88      0.52      0.65       346
       B-PER       0.92      0.75      0.82      1842

   micro avg       0.87      0.70      0.78      8603
   macro avg       0.87      0.67      0.75      8603
weighted avg       0.87      0.70      0.78      8603
```

Figure 12: Classification Report of NER tagging with MaxEnt without ’O’

```
In [38]: clf_ner.show_most_informative_features(15)
   5.254 +1:word.lower()=='gutters' and label is 'I-PER'
  -4.576 pos=='IN' and label is 'B-PER'
  -4.335 pos=='IN' and label is 'B-MISC'
   4.134 +1:word.lower()=='sundance' and label is 'I-MISC'
  -4.105 pos=='CD' and label is 'B-LOC'
   3.950 -1:word.lower()=='cassidy' and label is 'I-MISC'
   3.712 -1:word.lower()=='wisc' and label is 'I-LOC'
   3.712 -1:word.lower()=='colo' and label is 'I-LOC'
   3.685 +1:word.lower()=='chateaubriand' and label is 'I-PER'
  -3.383 pos=='CD' and label is 'B-PER'
   3.378 -1:word.lower()=='assoc' and label is 'I-ORG'
   3.378 -1:word.lower()=='fac' and label is 'I-ORG'
   3.373 +1:word.lower()=='witwatersrand' and label is 'I-ORG'
   3.087 -1:word.lower()=='fritz' and label is 'I-PER'
   3.069 -1:word.lower()=='azad' and label is 'I-LOC'
```

Figure 13: Most informative feature on NER model

7.2 Conditional Random Fields

In NER task, we continue seeing better results. It achieves a 0.88 f1 score on the test data excleuding 'O' tag.

```
In [9]: print(cm)
        |          B                   I                   |
        |    B     -     B     B     I     -     I     I       |
        |    -     M     -     -     -     M     -     -       |
        |    L     I     O     P     L     I     O     P       |
        |    O     S     R     E     O     S     R     E       |
        |    C     C     G     R     C     C     G     R     O |
--------+-------------------------------------------------------+
  B-LOC | <1598>  13    78    36     .     .     7     1   104 |
 B-MISC |    16 <760>   24    29     .     6     2     3    82 |
  B-ORG |    45    11 <1081>  100    1     .    16     2    85 |
  B-PER |    48     6    30 <1639>   .     2     9     7   101 |
  I-LOC |     2     .     1     . <206>    2    29    10     7 |
 I-MISC |     7    15     .     .     3 <252>   16    12    41 |
  I-ORG |     7     1     7     3    13    10 <610>   44    56 |
  I-PER |     3     .     4     7     7     4    20 <1234>  28 |
      O |    12     8    37    25     3    16    33     4<42621>|
--------+-------------------------------------------------------+
(row = reference; col = test)
```

Figure 14: Confusion Matrix of NER tagging with CRF

```
              precision    recall  f1-score   support

       B-LOC       0.92      0.87      0.89      1837
      B-MISC       0.93      0.82      0.88       922
       B-ORG       0.86      0.81      0.83      1341
       B-PER       0.89      0.89      0.89      1842
       I-LOC       0.88      0.80      0.84       257
      I-MISC       0.86      0.73      0.79       346
       I-ORG       0.82      0.81      0.82       751
       I-PER       0.94      0.94      0.94      1307
           O       0.99      1.00      0.99     42759

    accuracy                           0.97     51362
   macro avg       0.90      0.85      0.87     51362
weighted avg       0.97      0.97      0.97     51362

In [2]: print(metrics.flat_classification_report(y_test_ner, y_pred_ner_test,labels = labels))
              precision    recall  f1-score   support

       B-LOC       0.92      0.87      0.89      1837
       I-LOC       0.88      0.80      0.84       257
       B-PER       0.89      0.89      0.89      1842
       B-ORG       0.86      0.81      0.83      1341
       I-ORG       0.82      0.81      0.82       751
      B-MISC       0.93      0.82      0.88       922
      I-MISC       0.86      0.73      0.79       346
       I-PER       0.94      0.94      0.94      1307

   micro avg       0.90      0.86      0.88      8603
   macro avg       0.89      0.83      0.86      8603
weighted avg       0.90      0.86      0.88      8603
```

Figure 15: Classification Report of NER tagging with CRF

8 MaxEnt vs CRF

In general, CRF gives much better results. In the MaxEnt models above, I am showing the ones with using features from neighbour words, and upos in pos model, pos in chunking model and pos-chunk in ner model. In CRF, however, I achieve much higher results, without using neighbour features, because it captures much more information from the context anyway, and without upos-pos-chunk features. So, in feature-wise and accuracy-wise, CRF outperforms MaxEnt. Moreover its training time is significantly less than MaxEnt.

9 Conclusion

To conclude, I have experimented different features in different models. I have showed here the results I got, and conclude that CRF is performing better in all models with fewer features and less training time.