1 Feature Engineering

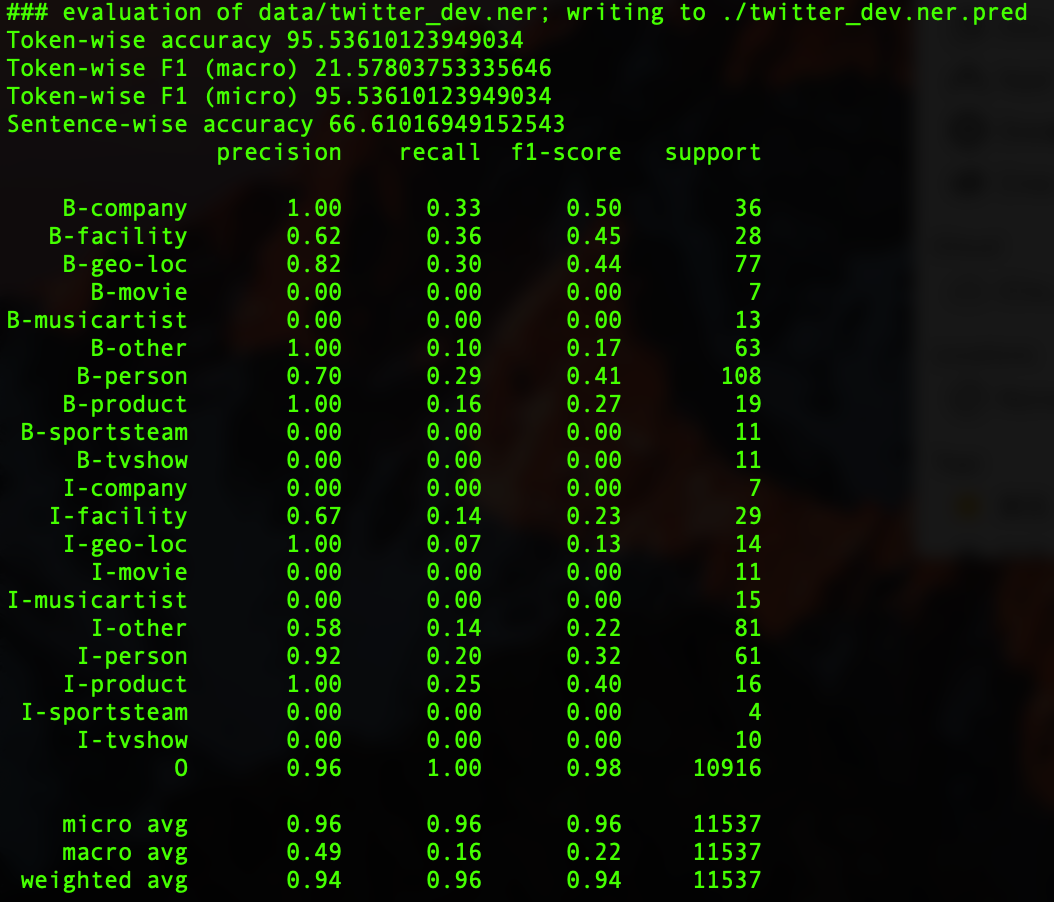
* 1. Feature design explanation

To begin with, I ran the default feature the script given and saved the data as follow:

**Twitter\_dev\_test.ner**



**Twitter\_dev.ner**



**The features I applied to improve F1 score as follow:**

First word

This feature finds out whether the current word is the first one of a sentence or not. In tweet, the first word usually is a named entity.

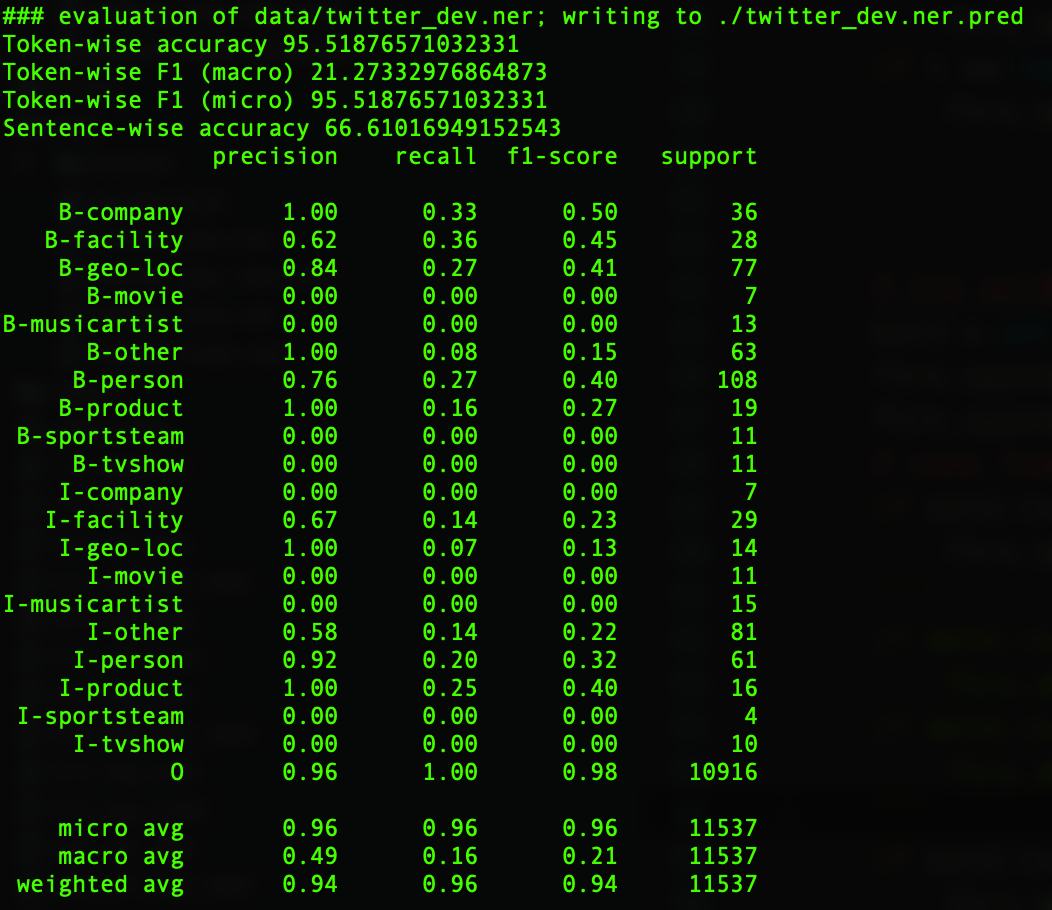
For example, Columbus B-geo-loc， according to this one, the first word ‘Columbus ’is a named entity.

Numeric

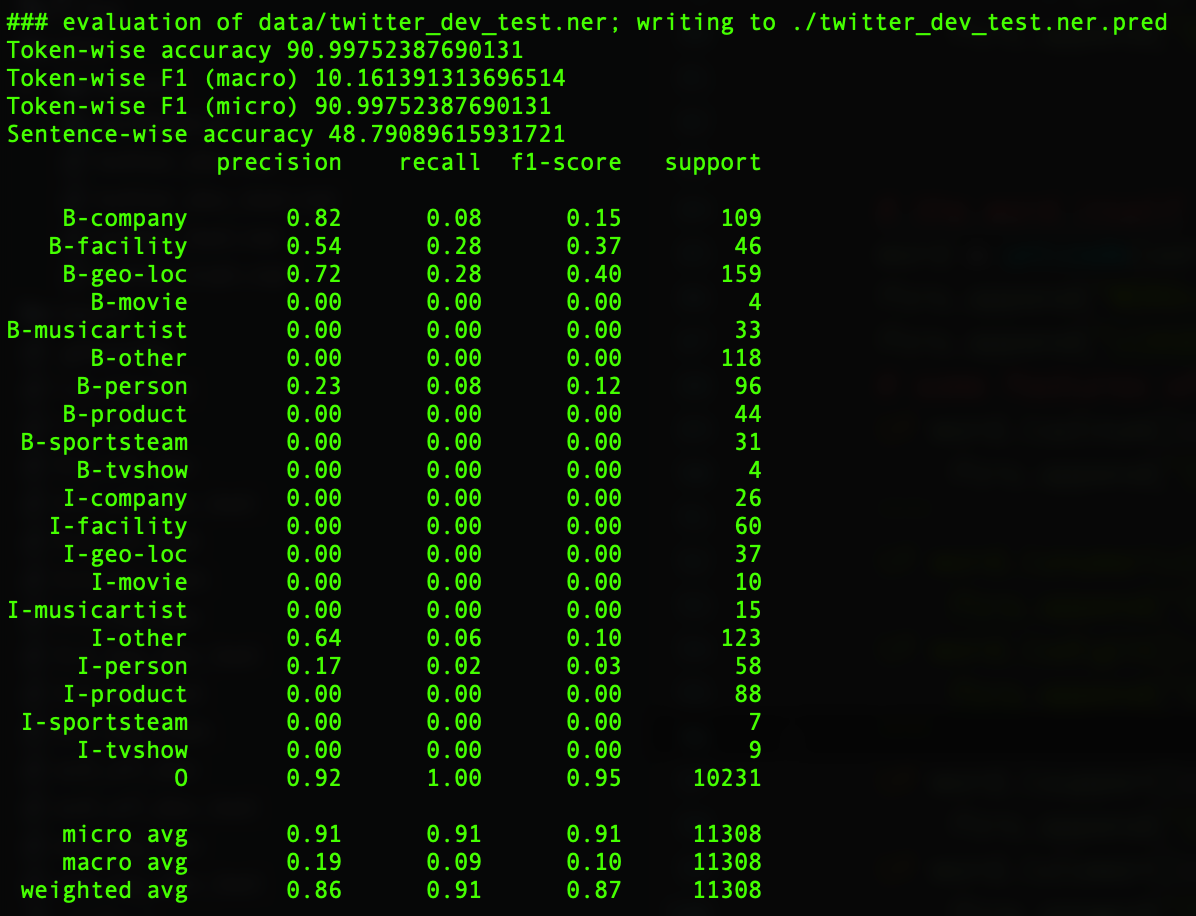
Determine whether current word is a number or not, usually number not a named entity but a count of named entity. For example, “a server 2008 (I-productd) R2”, “the number ”2008”belongs to named entity. Which is product type.

After I delete it from code the result as follow:

**Twitter\_dev\_test.ner**



**Twitter\_dev.ner**



Compared to the default result, without numeric result is worse, so numeric is an useful feature.

Previous word

By ad the previous word as a feature, this can make current word provides more information instead just about itself. For example, “a server 2008 (I-productd) R2” and the current word is ‘2008’, the previous word is ‘server’, which is a named entity.

Next word

By ad the next word as a feature, this can make current word provides more information instead just about itself. For example, “a server 2008 (I-productd) R2” and the current word is ‘2008’, the next word is ‘R2’, which is a named entity.

Upper case

Identify whether current word is in uppercase or not, usually if a word is in uppercase it should be a named entity. Contrast to lowercase feature this one is helpful to tweets. But its not reliable due lots of tweets still not follow this rule.

For example, ‘My Oakland (B-geo-loc) line’, in this tweet there are two fully uppercase words but actually only ‘Oakland’ is named entity instead of ‘My’.

Lower case

Identify whether current word is in lowercase or not, usually if a word is in lowercase its not a named entity.

First word.

This feature finds out whether the current word is the first one of a sentence or not. In tweet, the first word usually is a named entity.

For example, ‘Pedigree (B-company) Donates Dog’, according to this one, the first word ‘Pedigree ’is a named entity.

Last word.

This feature finds out whether the current word is the last one of a sentence or not. In context the last word usually is a named entity. After applied this feature, based on python script evaluation I get follow data:

twitter\_dev.ner:

|  |  |  |
| --- | --- | --- |
|  | basic | advanced |
| accuracy | 95.54 | 95.67 |
| F1(macro) | 21.58 | 22.85 |
| F1(micro) | 95.54 | 95.67 |

twitter\_dev\_test.ner:

|  |  |  |
| --- | --- | --- |
|  | basic | advanced |
| accuracy | 91.02 | 91.52 |
| F1(macro) | 10.92 | 17.18 |
| F1(micro) | 91.02 | 91.52 |

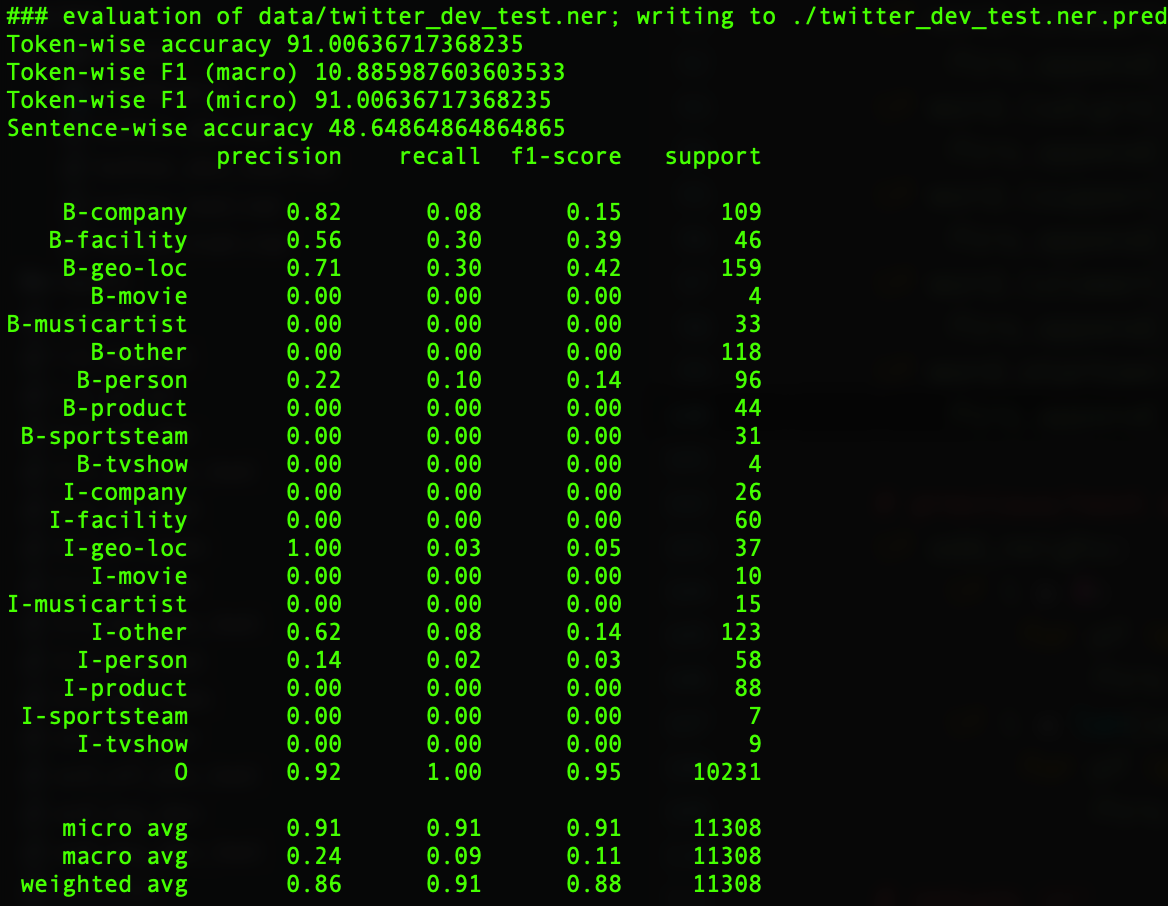
Stop word

This feature is the set of English words that usually appear at the end of an English sentence. These words are not named entities but they are useful to identify named entity.

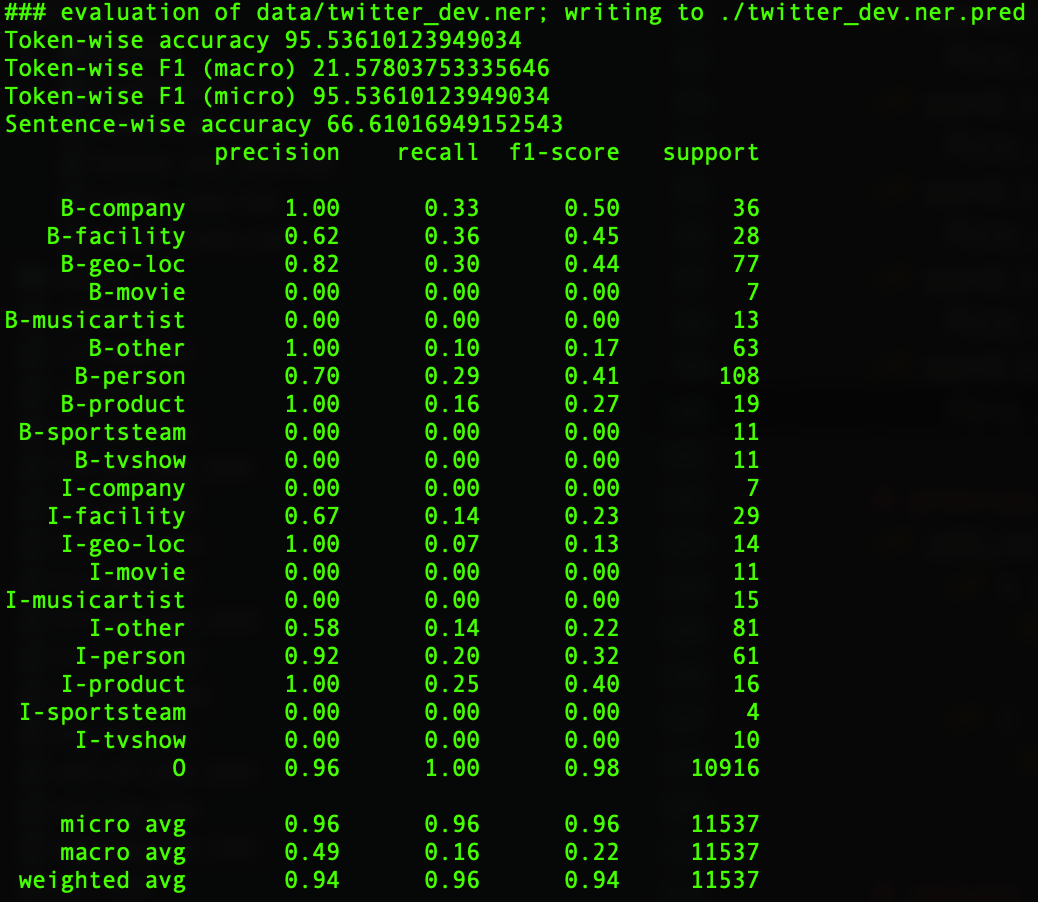
IS\_URL

This feature is used to identify whether the current is a web link or not by the key word ’http , one web link is not a named entity. After adding hyperlink, the result as follow:

**Twitter\_dev\_text.ner**



**Twitter\_dev.ner**



By comparing this result and default, it seems no change at all, but by combine it with lexicon and other useful features, I think it still usefull.

Lexicon word

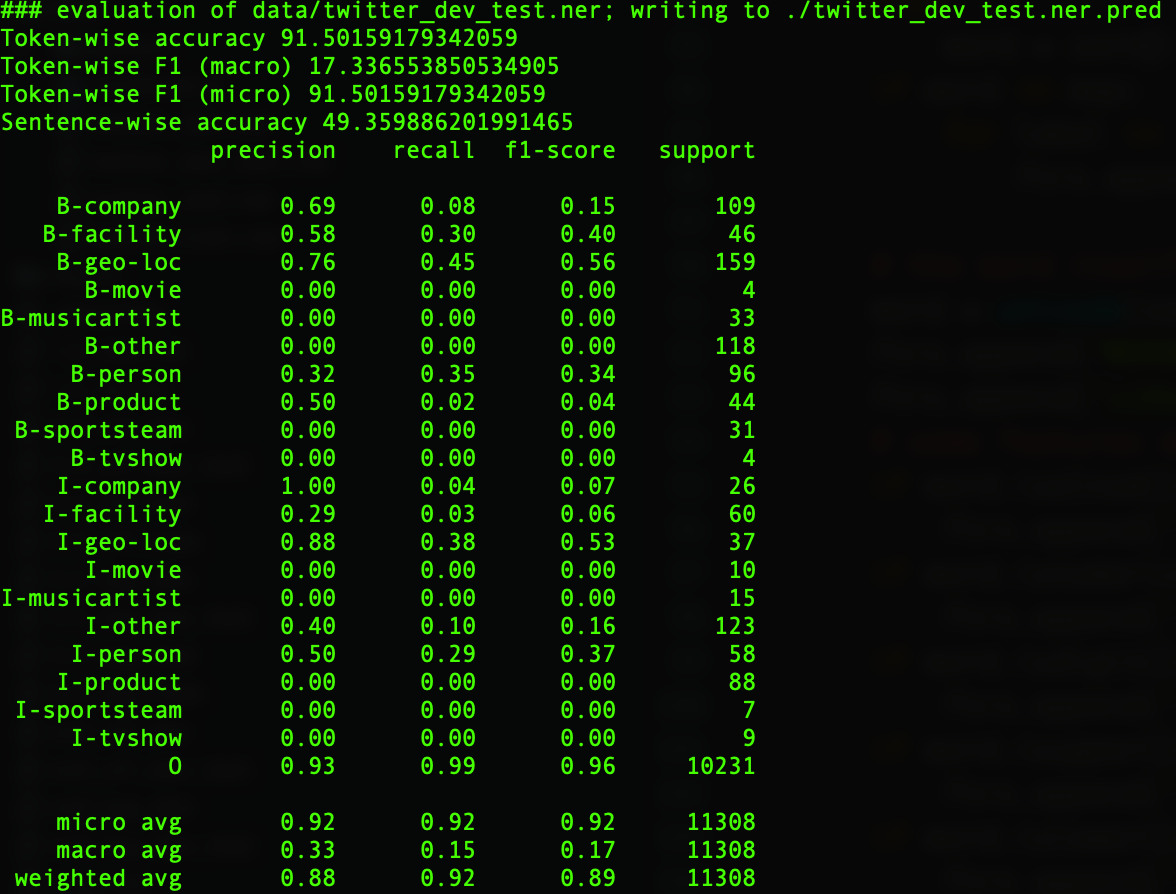
These features includes some sets I added to the code as follow:

Location; Business Product; Product; Tv show; Sport team; Location; Name;

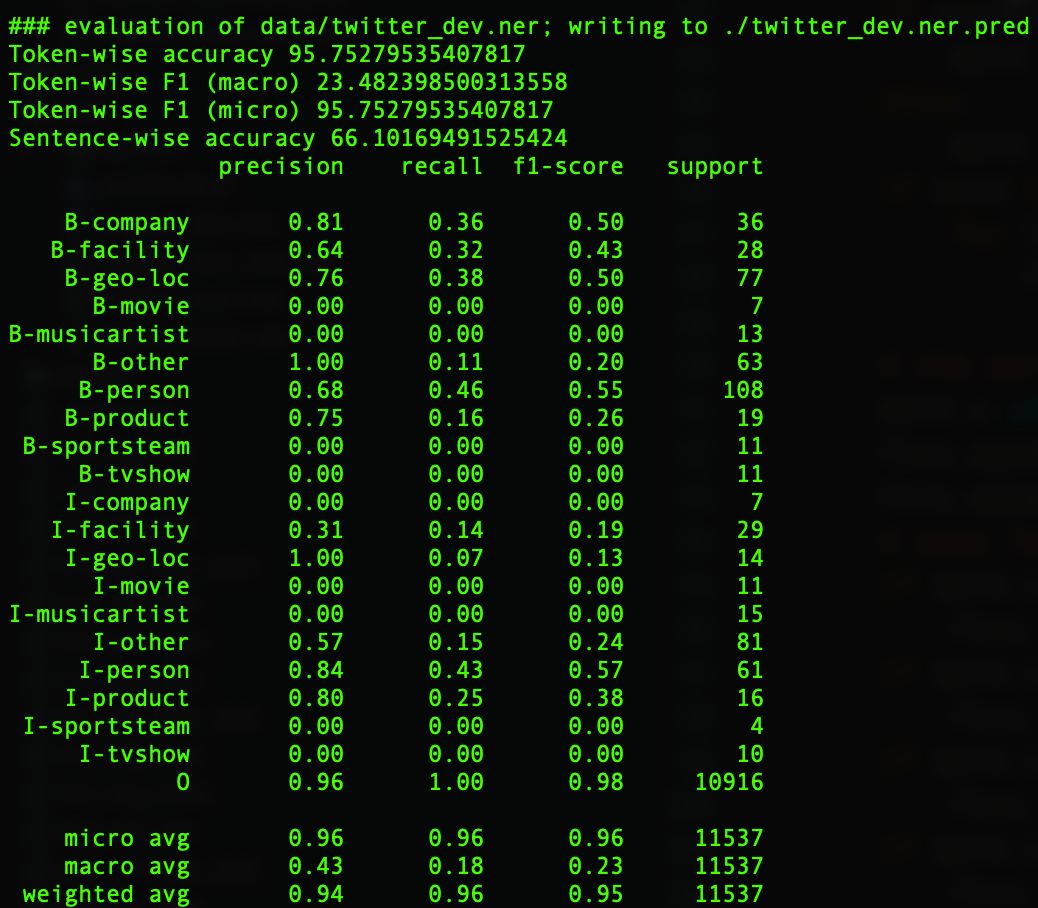
The label ‘name’ Including ‘last name' and ‘first name’ two features, these two features contribute to name, which is named entity.

For example, “LG Venus” in the set ‘product’ and ‘business\_consumer\_product’. so both label ‘Product’ and ‘Business Product’ is attached and others so on.

**twitter\_dev\_test.ner:**



**twitter\_dev.ner:**

****

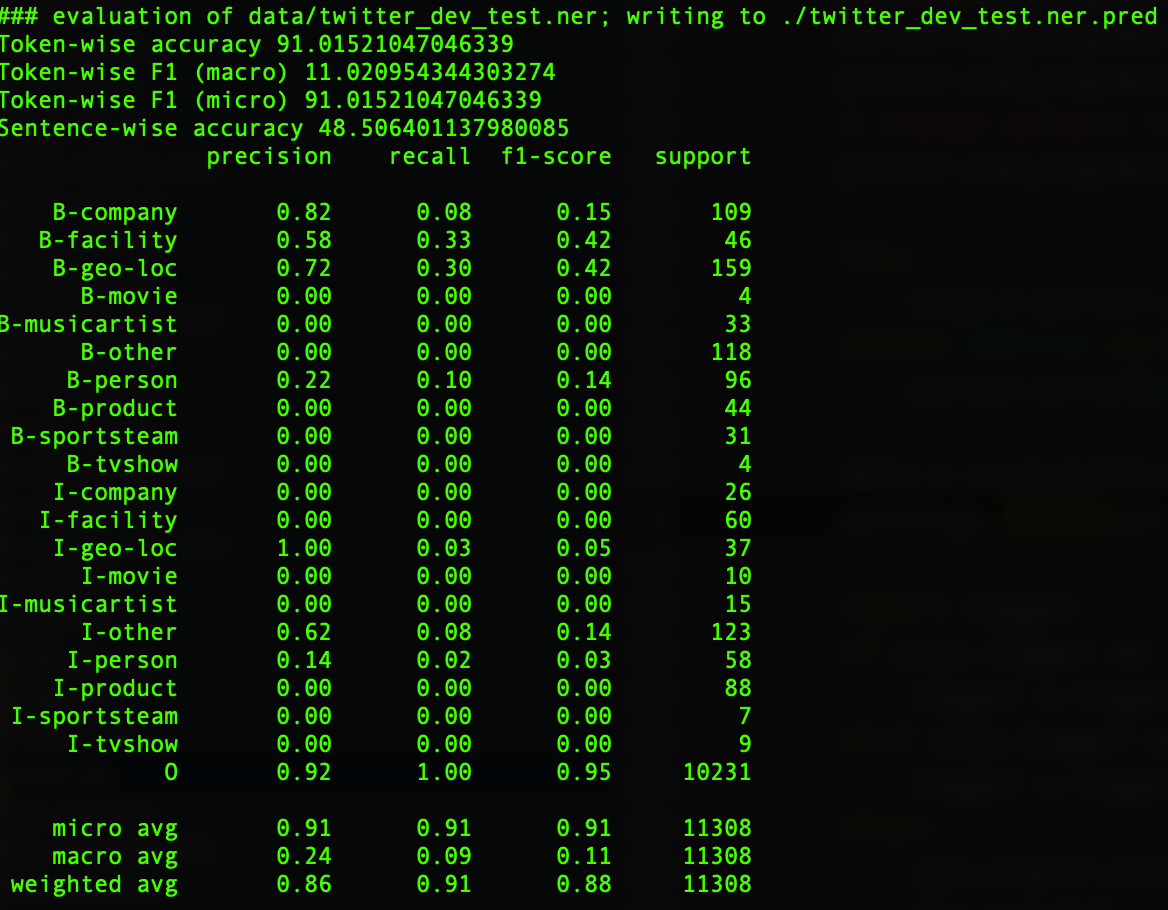
Its obvious that after applied lexicon features the result (especially F1) improved a lot.

**The features I applied not improve F1 score as follow:**

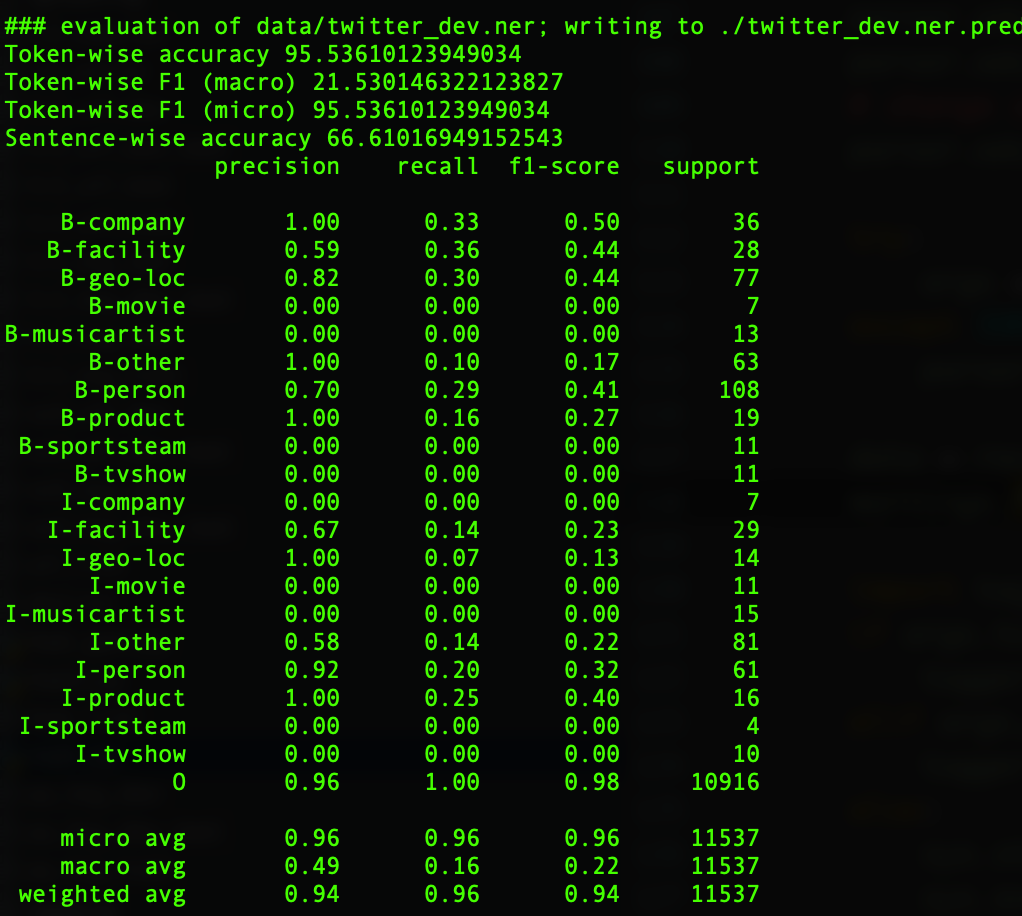
First word is number

Counting the number appear at the beginning of sentence, which usually are the pre part of named entity. But this feature is not work well. According to the follow data, the result gets worse.

**twitter\_dev\_test.ner:**

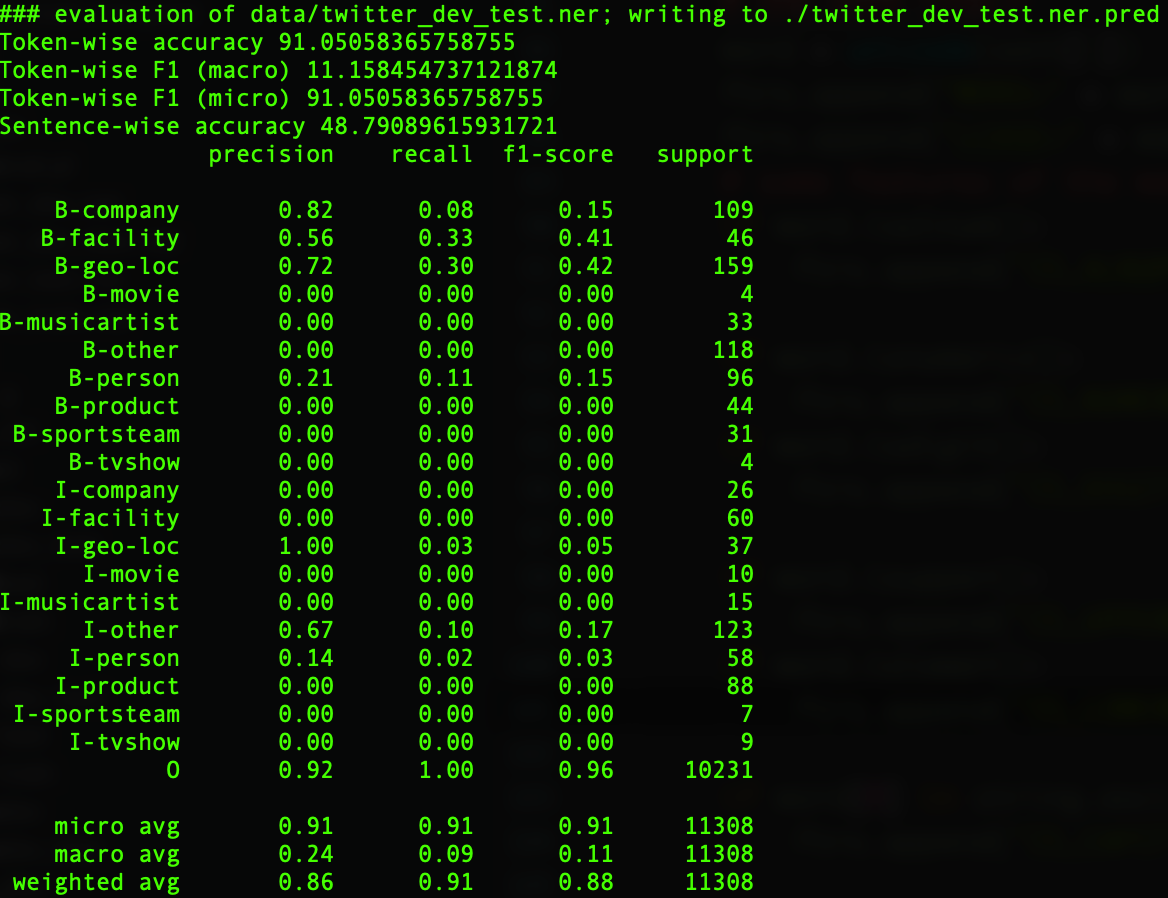
****

**twitter\_dev.ner:**

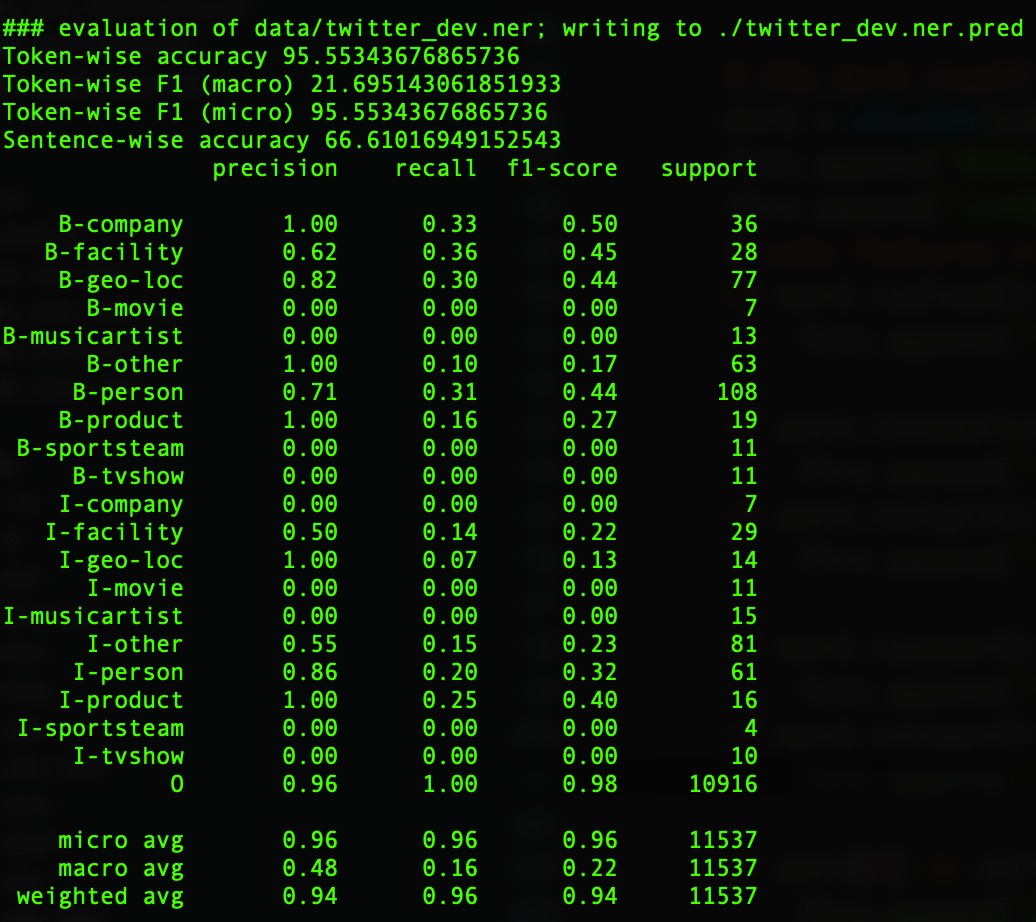
****

First character is uppercase

**twitter\_dev\_test.ner:**

****

**twitter\_dev.ner:**



this result makes no difference with the default’s , so it is not useful.

* 1. Implementation of additional features

The features I applied are the features useful talked in section 1.1.

The improvement as follow:

According the results, its obvious my improvement worked both Macro F1 scoreand Micro F1 score. To be specifically, the Macro F1 score improved from 21.57 to 23.51. The Micro F1 score improved from 95.53 to 95.76

The original code’s result

### evaluation of data/twitter\_dev.ner; writing to ./twitter\_dev.ner.pred  
Token-wise accuracy 95.53610123949034  
Token-wise F1 (macro) 21.57803753335646  
Token-wise F1 (micro) 95.53610123949034  
Sentence-wise accuracy 66.61016949152543  
 precision recall f1-score support  
  
 B-company 1.00 0.33 0.50 36  
 B-facility 0.62 0.36 0.45 28  
 B-geo-loc 0.82 0.30 0.44 77  
 B-movie 0.00 0.00 0.00 7  
B-musicartist 0.00 0.00 0.00 13  
 B-other 1.00 0.10 0.17 63  
 B-person 0.70 0.29 0.41 108  
 B-product 1.00 0.16 0.27 19  
 B-sportsteam 0.00 0.00 0.00 11  
 B-tvshow 0.00 0.00 0.00 11  
 I-company 0.00 0.00 0.00 7  
 I-facility 0.67 0.14 0.23 29  
 I-geo-loc 1.00 0.07 0.13 14  
 I-movie 0.00 0.00 0.00 11  
I-musicartist 0.00 0.00 0.00 15  
 I-other 0.58 0.14 0.22 81  
 I-person 0.92 0.20 0.32 61  
 I-product 1.00 0.25 0.40 16  
 I-sportsteam 0.00 0.00 0.00 4  
 I-tvshow 0.00 0.00 0.00 10  
 O 0.96 1.00 0.98 10916  
  
 micro avg 0.96 0.96 0.96 11537  
 macro avg 0.49 0.16 0.22 11537  
 weighted avg 0.94 0.96 0.94 11537

My improvement’s result:

### evaluation of data/twitter\_dev.ner; writing to ./twitter\_dev.ner.pred  
Token-wise accuracy 95.7614631186617  
Token-wise F1 (macro) 23.514567880578184  
Token-wise F1 (micro) 95.7614631186617  
Sentence-wise accuracy 66.27118644067797  
 precision recall f1-score support  
  
 B-company 0.81 0.36 0.50 36  
 B-facility 0.64 0.32 0.43 28  
 B-geo-loc 0.78 0.38 0.51 77  
 B-movie 0.00 0.00 0.00 7  
B-musicartist 0.00 0.00 0.00 13  
 B-other 1.00 0.11 0.20 63  
 B-person 0.68 0.46 0.55 108  
 B-product 0.75 0.16 0.26 19  
 B-sportsteam 0.00 0.00 0.00 11  
 B-tvshow 0.00 0.00 0.00 11  
 I-company 0.00 0.00 0.00 7  
 I-facility 0.31 0.14 0.19 29  
 I-geo-loc 1.00 0.07 0.13 14  
 I-movie 0.00 0.00 0.00 11  
I-musicartist 0.00 0.00 0.00 15  
 I-other 0.60 0.15 0.24 81  
 I-person 0.84 0.43 0.57 61  
 I-product 0.80 0.25 0.38 16  
 I-sportsteam 0.00 0.00 0.00 4  
 I-tvshow 0.00 0.00 0.00 10  
 O 0.96 1.00 0.98 10916  
  
 micro avg 0.96 0.96 0.96 11537  
 macro avg 0.44 0.18 0.24 11537  
 weighted avg 0.94 0.96 0.95 11537

2 Viterbi coding

2.1 Pseudocode for Viterbi algorithm

The pseudocode for the Viterbi algorithm is as follows:

‘L’ is the number of labels

‘N’ is the number of tokens of a sentence

‘start’ is the set of start probabilities, in the same order of the labels

‘end ’ is the set of end probabilities, in the same order of the labels

‘table’ be the matrix to reserve the max probability value of tag sequence

‘backpointer’ be the matrix to reserve the index of back pointers

// Initialization: the start probabilities of first column

for i (0..L-1):

table[i][0] = start[i] + emission[0][i]

// Recursion: all other columns, calculate the best values

for j (1, N-1):

for i (0, L-1):

s = -(MAX\_INT-1)

for k (0, L-1):

temp = list((table[k][j - 1] + trans[k][i]))

s = emission[j][i] + table[k][j - 1] + trans[k][i]

if s> temp

table[i][j] = s

backpointer[i][j] = k

// Last Column: add the end probabilities of last column

for i (0, L-1):

end = table[i][N - 1] + end[i]

last\_seq = np.argmax(end)

score = max(end\_prob)

// Construct best sequence, initial s and index minor than 0.

s = -sys.maxint -1

index = -1

//find the best sequence

for i (0, L-1):

    if table[i][-1] >s:

        s = table[i][-1]

        index = i

while(I = N-1; i != -1; i--):

    sequence = [index] + y

    index = back[index][i]

return score, sequence

3 Compare Logistic Regression to CRFs

3.1 Analytics

I think CRF will do the work of sequence tagging better than logistic regression method.

1. CRF is a discriminative model, which is easier to use due it does not need features independent assumption. Since CRF is a discriminative model, it directly uses conditional probability instead of joint probability.
2. CRF use global normalization instead of local normalization, which can give a better result for general.
3. CRF is a feature based model and we can do a lot of improvement by modify features.

On the other hand, logistic regression need do more things because it converting the sequence tagging problem into independent classification problem. The independent classification problem needs calculate joint probabilities at first, that will require independent assumption which will increase the difficulties.

At all, CRF is the better choice to handle this problem than Logistic regression.

For example:

Waco B-other  
PD I-other  
was O  
on O  
scene O  
before O  
the O  
shooting O  
started O  
because O  
officers O  
knew O  
the O  
gangs O  
were O  
going O  
to O  
meet O  
. O  
Police O  
say O  
this O  
prevented O  
more O  
deaths O

The result of logistic regression is:

Waco B-other O  
PD I-other O  
was O O  
on O O  
scene O O  
before O O  
the O O  
shooting O O  
started O O  
because O O  
officers O O  
knew O O  
the O O  
gangs O O  
were O O  
going O O  
to O O  
meet O O  
. O O  
Police O O  
say O O  
this O O  
prevented O O  
more O O  
deaths O O

The result of CRF is:

Waco B-other B-sportsteam  
PD I-other I-sportsteam  
was O O  
on O O  
scene O O  
before O O  
the O O  
shooting O O  
started O O  
because O O  
officers O O  
knew O O  
the O O  
gangs O O  
were O O  
going O O  
to O O  
meet O O  
. O O  
Police O O  
say O O  
this O O  
prevented O O  
more O O  
deaths O O

* 1. Experiment

Applied CONLL evaluation.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | BASIC FEATURES | | ADVANCED FEATURES | |
|  |  | LogReg | CRF | LogReg | CRF |
| Twitter\_dev.ner.pred | Accuracy | 95.54 | 95.77 | 95.76 | 96.07 |
| Precision | 49.61 | 60.61 | 48.9 | 62.50 |
| Recall | 16.89 | 26.81 | 23.86 | 33.51 |
| FB1 | 25.20 | 37.17 | 32.07 | 43.63 |
| Twitter\_dev\_test.ner.pred | Accuracy | 91.02 | 91.31 | 91.50 | 91.71 |
| Precision | 32.35 | 46.82 | 31.89 | 44.51 |
| Recall | 8.54 | 15.99 | 14.91 | 23.29 |
| FB1 | 13.51 | 23.84 | 20.32 | 30.58 |

According to the sheet, the result of advanced is higher both logistic regression and crf method. Compared CRF and logreg, the CRF method has better result and higher FB1 score, that indicate CRF is a better method which can give higher score, due to crf method based on conditional probability and it is a discriminative model. Let’s focus on FB1 score, which is significant to evaluate the features are good or not.

For witter\_dev.ner, it improved 6.8 in average.

For witter\_dev\_test.ner, it improved 6.7 in average.

That mainly due to the advanced one gives model more features and improved model’s behavior.

Applied python script evaluation.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | BASIC FEATURES | | ADVANCED FEATURES | |
|  |  | LogReg | CRF | LogReg | CRF |
| Twitter\_dev.ner.pred | Token-wise accuracy | 95.54 | 95.77 | 95.76 | 96.07 |
| Token-wise F1 f(macro) | 21.58 | 29.56 | 23.51 | 29.29 |
| Token-wise F1 (micro) | 95.54 | 95.77 | 95.76 | 96.07 |
| Sentence-wise accuracy | 66.61 | 68.64 | 66.27 | 68.64 |
| Twitter\_dev\_test.ner,pred | Token-wise accuracy | 91.02 | 91.31 | 91.50 | 91.71 |
| Token-wise F1 (macro) | 10.92 | 17.98 | 17.36 | 23.45 |
| Token-wise F1 (micro | 91.02 | 91.31 | 91.50 | 91.71 |
| Sentence-wise accuracy | 48.65 | 50.50 | 49.36 | 52.20 |

According to the python script evaluation, advanced feature has better accuracy, I prefer analyzing F1(micro) because it can analyze the data in general better, macro will calculate F1 score of each class and then do average, which is not useful.

4 Evaluation Metrics

python script evaluation

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | BASIC FEATURES | | ADVANCED FEATURES | |
|  |  | LogReg | CRF | LogReg | CRF |
| Twitter\_dev.ner.pred | Token-wise accuracy | 95.54 | 95.77 | 95.76 | 96.07 |
| Token-wise F1 f(macro) | 21.58 | 29.56 | 23.51 | 29.29 |
| Token-wise F1 (micro) | 95.54 | 95.77 | 95.76 | 96.07 |
| Sentence-wise accuracy | 66.61 | 68.64 | 66.27 | 68.64 |
| Twitter\_dev\_test.ner,pred | Token-wise accuracy | 91.02 | 91.31 | 91.50 | 91.71 |
| Token-wise F1 (macro) | 10.92 | 17.98 | 17.36 | 23.45 |
| Token-wise F1 (micro | 91.02 | 91.31 | 91.50 | 91.71 |
| Sentence-wise accuracy | 48.65 | 50.50 | 49.36 | 52.20 |

Conll script evaluation

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | BASIC FEATURES | | ADVANCED FEATURES | |
|  |  | LogReg | CRF | LogReg | CRF |
| Twitter\_dev.ner.pred | Accuracy | 95.54 | 95.77 | 95.76 | 96.07 |
| Precision | 49.61 | 60.61 | 48.9 | 62.50 |
| Recall | 16.89 | 26.81 | 23.86 | 33.51 |
| FB1 | 25.20 | 37.17 | 32.07 | 43.63 |
| Twitter\_dev\_test.ner.pred | Accuracy | 91.02 | 91.31 | 91.50 | 91.71 |
| Precision | 32.35 | 46.82 | 31.89 | 44.51 |
| Recall | 8.54 | 15.99 | 14.91 | 23.29 |
| FB1 | 13.51 | 23.84 | 20.32 | 30.58 |

First of all, both methods are useful to do evaluation job. CONNL and python evaluation can get good results for this application.

About the differences,

The CONLL is a chunk evaluation method and its result includes NER Accuracy, Precision, Recall, FB1 for all classes and also given the same information for every class.

the python evaluation can do feature engineering and I can add any feature I interested. The result includes token wise accuracy, token wise F1 (micro), token wise F1 (macro) and sentence wise accuracy are got for all classes. What’s more, the result also has every class’s recall, precision and F1 (including B- and I- tags). Due to more useful and detail data given by the python evaluation script, I think that is better.