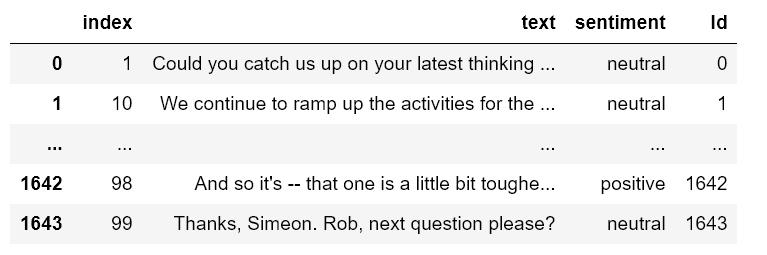
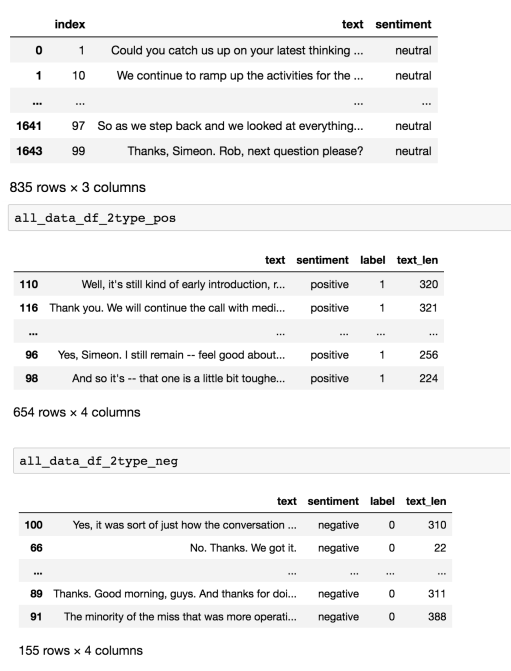
Homework 3 for NLP

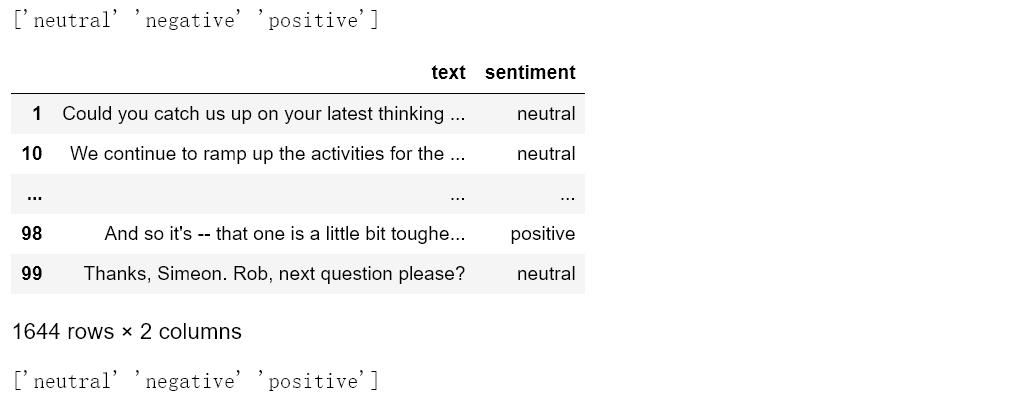
Wenjun song, Haimin Zhang, Yanjun Lyu

Before we build the models, we should deal with the data from 12 teams, and combine them together. When we combined the data, there are different kinds of errors for the “.json” files, and most of them are about String problems which are the typical problems for such files. So in the future when we deal with these kinds of files, such as “json” and “txt”, we should be careful about the string format.

For better indexing the information of the file, we added a “Id” colunm:



Then we wrangled data into a proper format to train a NLP model in the next steps. During this process, we found many errors of the 12 files mentioned above. So we obtain the final dataset used in the following steps:

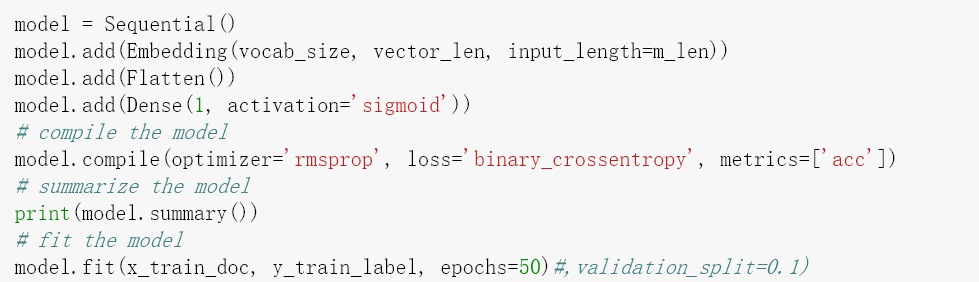


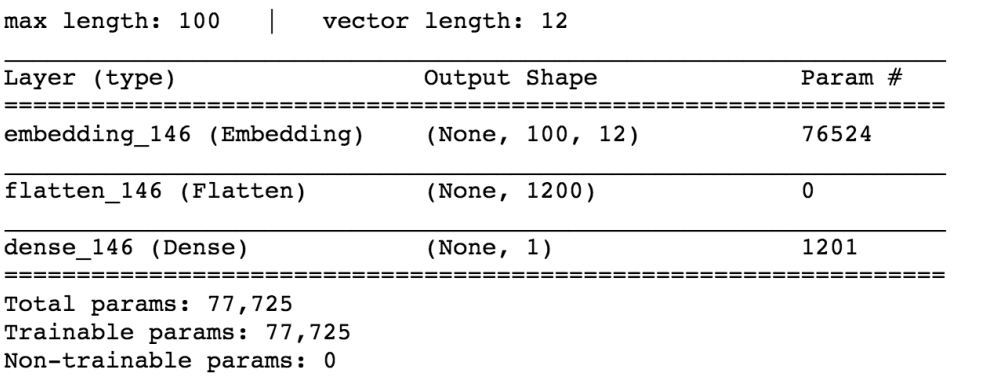
|  |  |  |
| --- | --- | --- |
| sentiment | number | Percent |
| Neutral | 835 | 0.507907542 |
| Positive | 654 | 0.397810218 |
| Negative | 155 | 0.094282238 |

Experience 1:

For GLOVE model:

We set such a model:

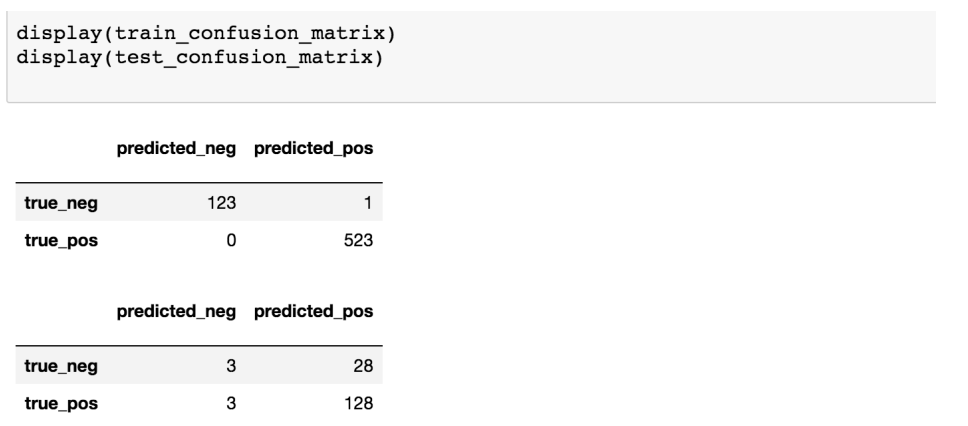




This is the output of Training Accuracy and Test Accuracy:



It is possible that this model is not the best model, but it is the relative best  
one among some parameters combinations. we have the confusion  
matrix for it like this:



So this model is not bad, the test accuracy is 0.8086.

For RNN model:

We build a SimpleRNN model to try to extract the connection between words in the documents. In this model, each word first obtains a feature vector from the embedding layer. Then, we further encode the feature sequence using a simple recurrent neural network to obtain sequence information. Finally, we transform the encoded sequence information to output through the fully connected layer, for this task, the last layer is a softmax function.

The structure of model:

model2.summary()

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

embedding\_1 (Embedding) (None, None, 32) 194880

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

simple\_rnn\_1 (SimpleRNN) (None, None, 32) 2080

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

simple\_rnn\_2 (SimpleRNN) (None, None, 32) 2080

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

simple\_rnn\_3 (SimpleRNN) (None, 32) 2080

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_1 (Dense) (None, 3) 99

=================================================================

Total params: 201,219

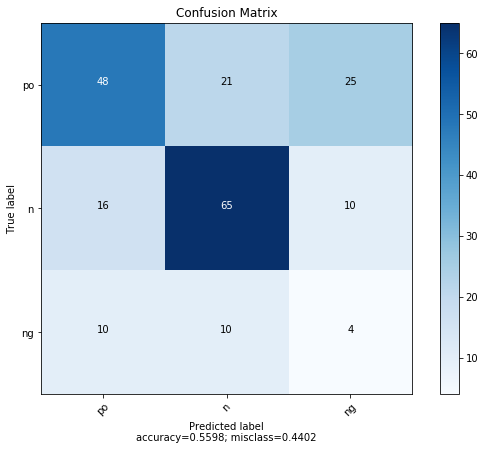
Trainable params: 201,219

Non-trainable params: 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

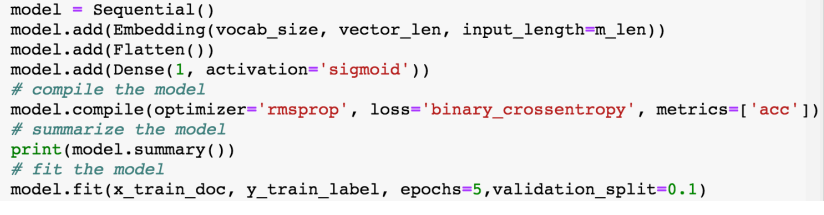
The accuracy come to 0.5598 for test set.

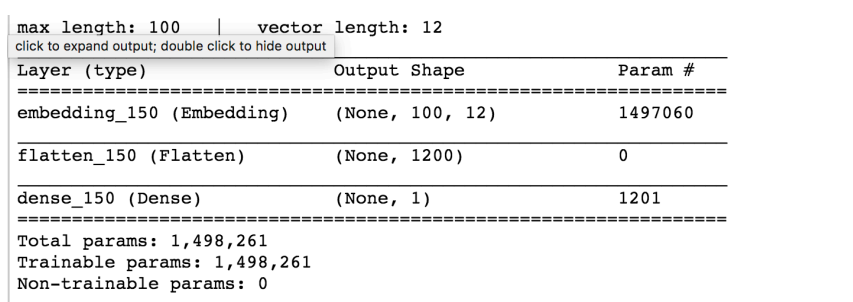
The confusion matrix as below:



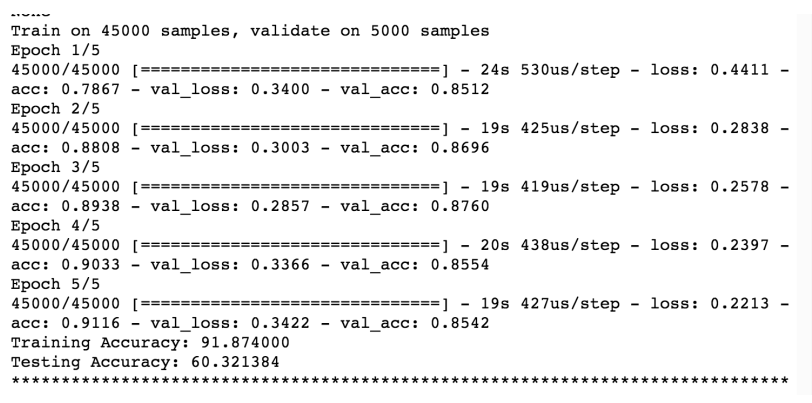
Experiment 2:

After wrangling the movie data, we used the model of experiment 1 to test  
the movie data.the model is like this:

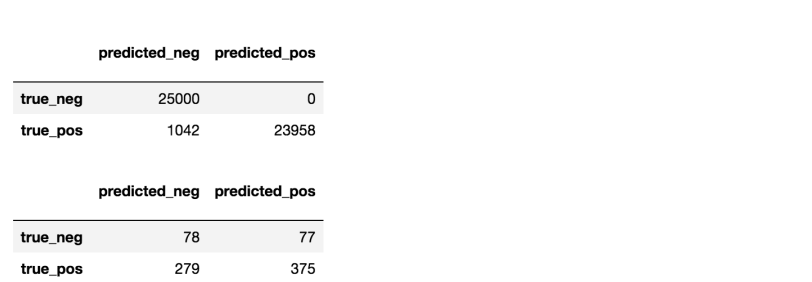




This is the output when we set epoch=5, or else it is easily to overfit.



The confusion matrix is:

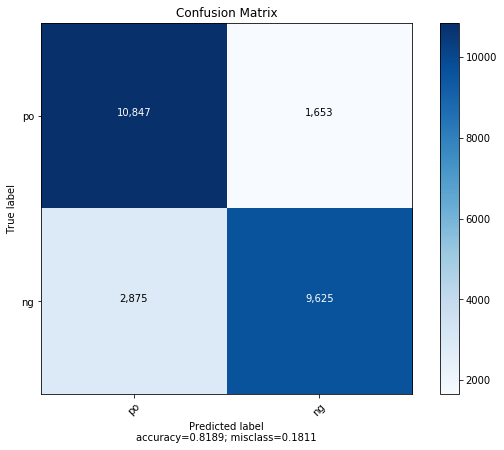
  
For the output, for validation accuracy is good, which means out model is  
good for this dataset. But for test data, the accuracy - 0.6032 is less than the  
validation. Maybe it is because the difference of test data and other data,  
but overall the model is not bad for us.

For RNN model:

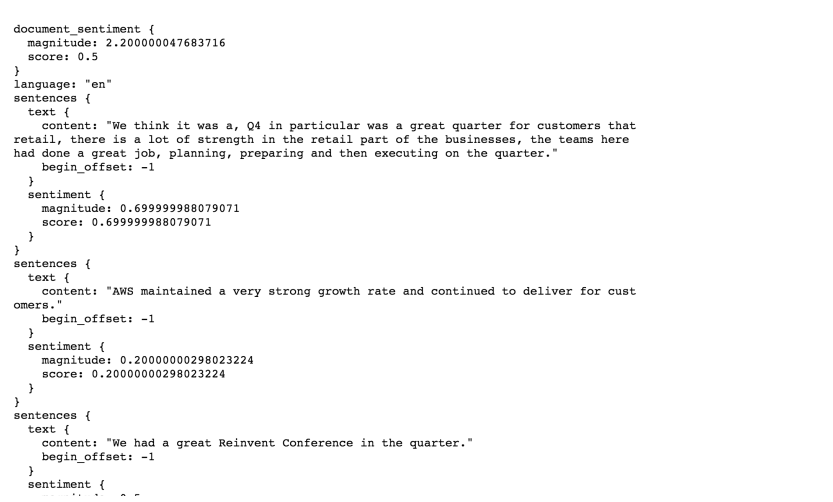
The model structure is almost like the RNN model used in experiment 1, the different is last layer is sigmoid function.

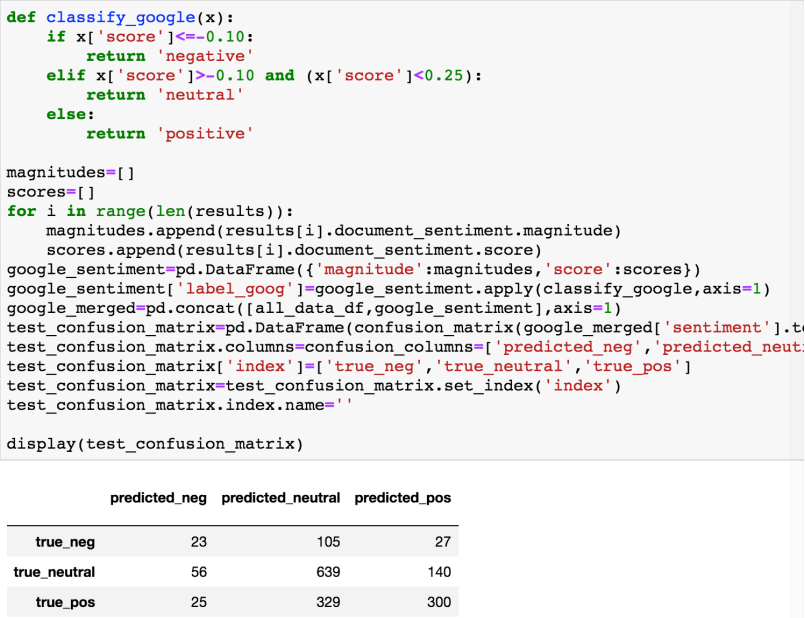
We use the IMDB movie review dataset from Keras.

The accuracy come to 0.8189 for test set.

The confusion matrix as below:

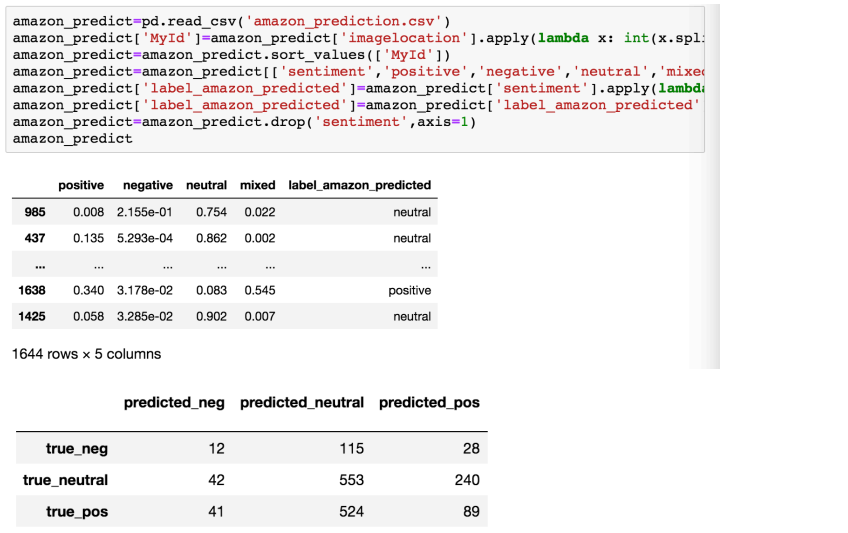
Experiment 3:  
We firstly used Google and Amazon API to test the data:  
**This is the output of google:**





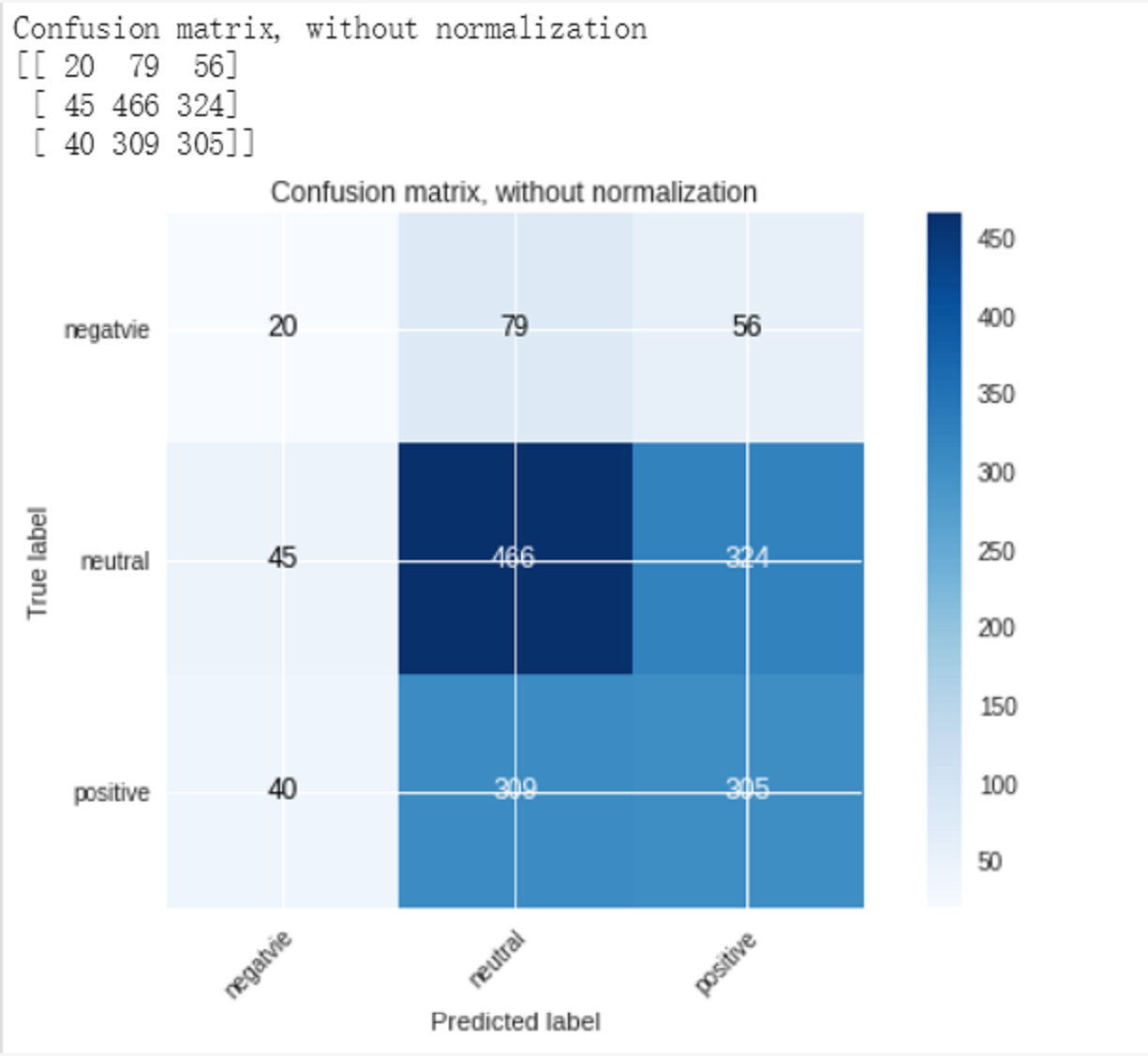
We can see the accuracy is 58.51

**The output of Amazon is :**



The accuracy is 39.78.

**The output of Microsoft is:**



The accuracy is 0.4811.

Unfortunately, we had spent more than 10 hours to handle IBM’s API, but we did’t clean up the bugs, so we haven’t get the conclusion of that method.

We just made such a step:

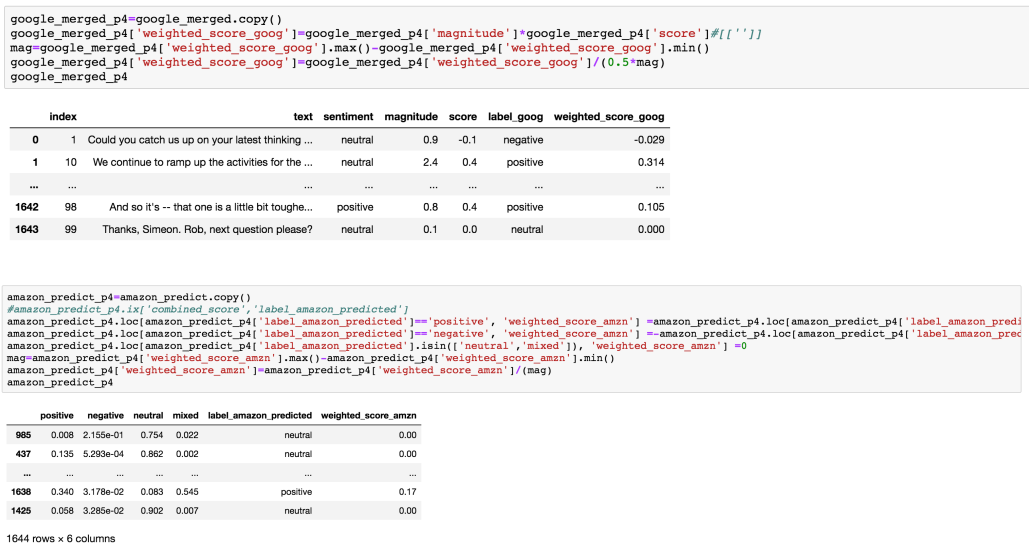


|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | BOW | GLOVE | RNN | GOOGLE | AMAZON | MICROSOFT |
| Test  Accuracy | 0.3171 | - | 0.5598 | 0.5851 | 0.3978 | 0.4811 |

Comparing these three API models with our models, we can see the Google API is the best one.

Experiment 4:

And then we merged the Google and Amazon together. Due to the format is different, so we should firstly change the data format into the same format:

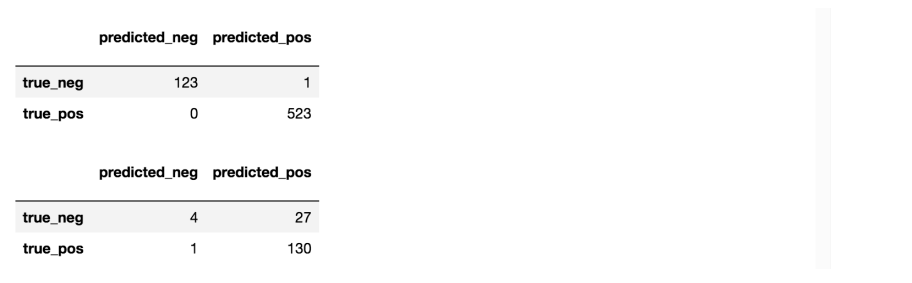


We set a cutoff to identify the three classes as following and get the  
confusion matrix:



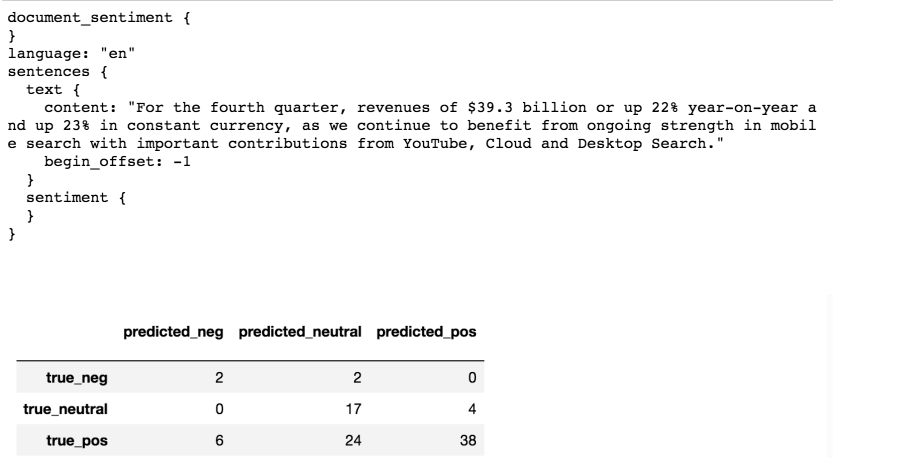
For this situation, the merged model’s accuracy is: 0.4779, which is better than Amazon, and less than Google.

From the conclusions above, we can see the google is the best model for these data, so we use google API to test the GE file and other teams files, and then get the confusion matrix.

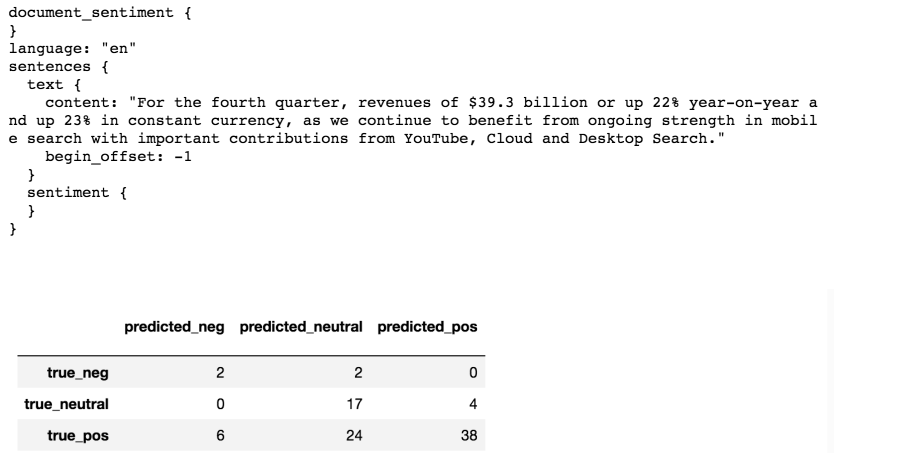


**The following is the confusion matrix for the 12 files from other teams:**

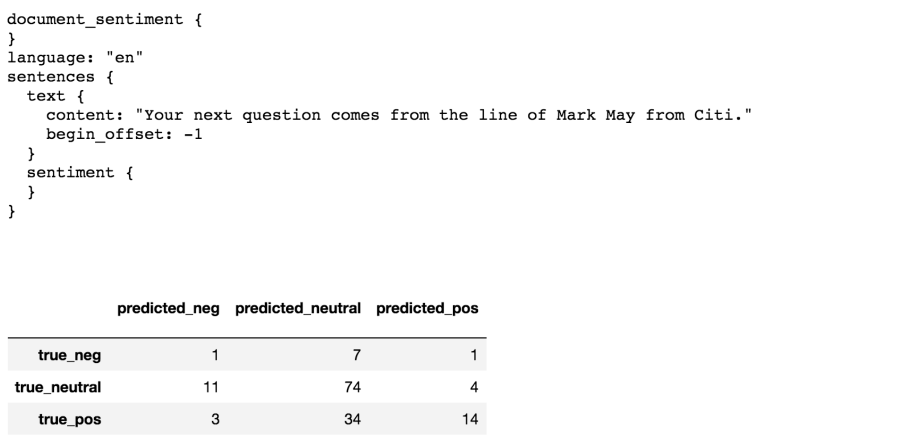
**1. Team1\_Google:**



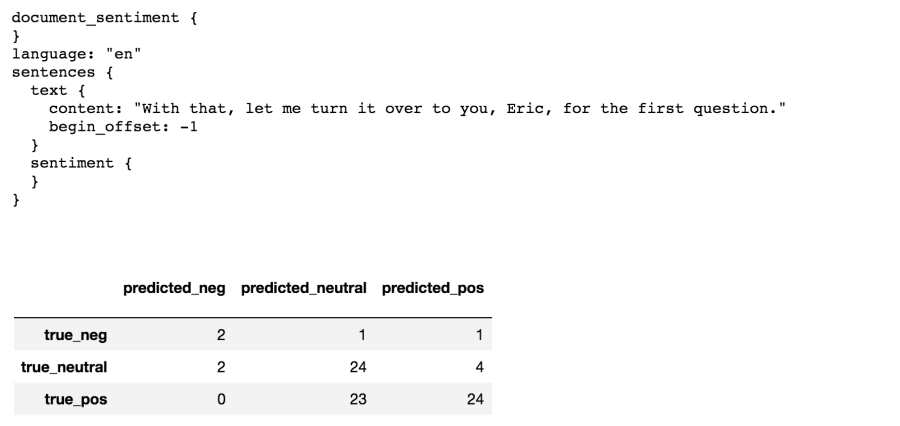
**2. Team2\_Amazon**



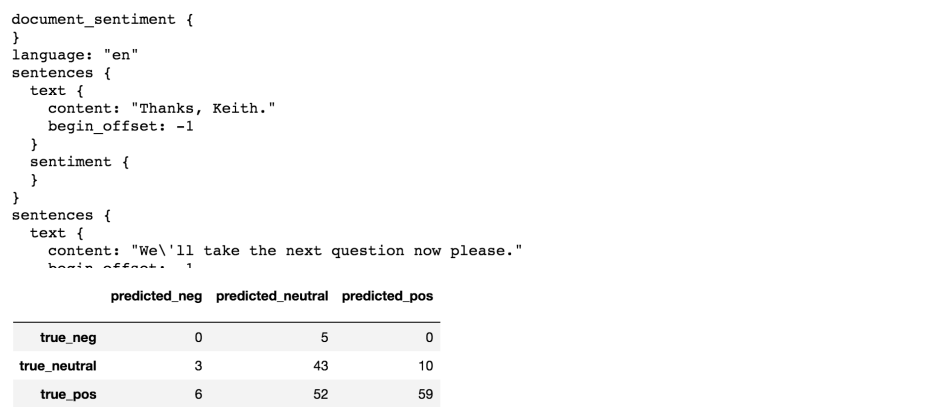
**3. Team3\_Facebook**



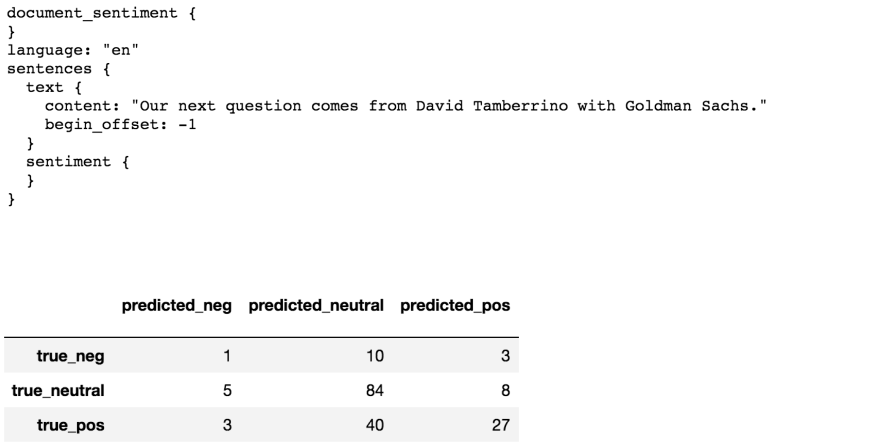
**4.** . **Team4\_Netflix**



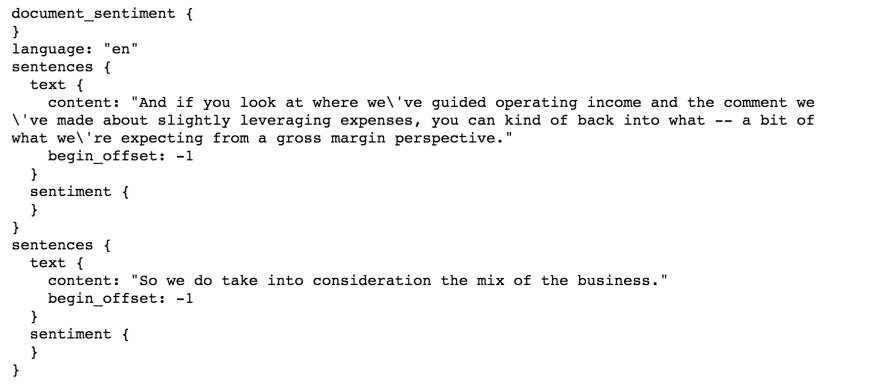
**5. Team5\_Microsoft**

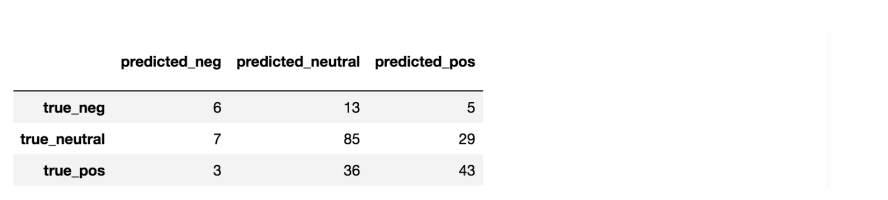


**6. Team6\_Tesla**

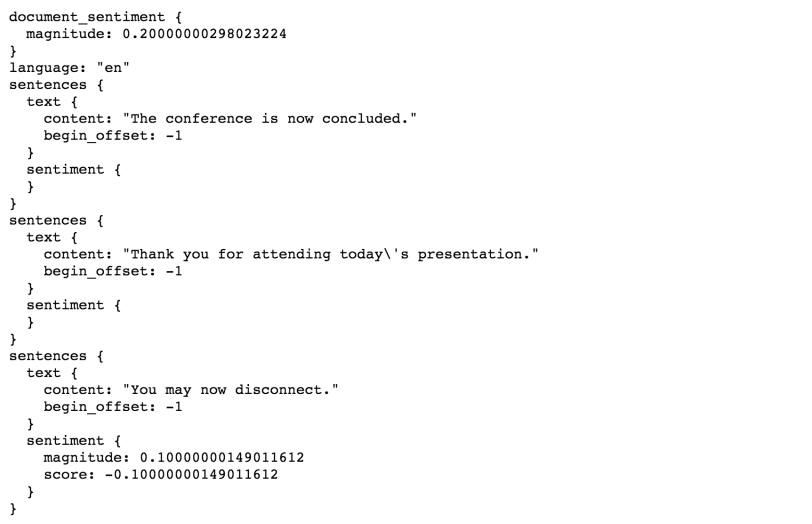


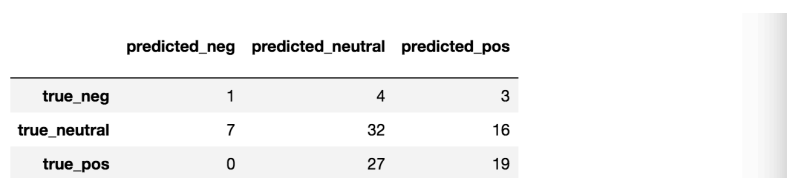
**7. Team7\_Walmart**



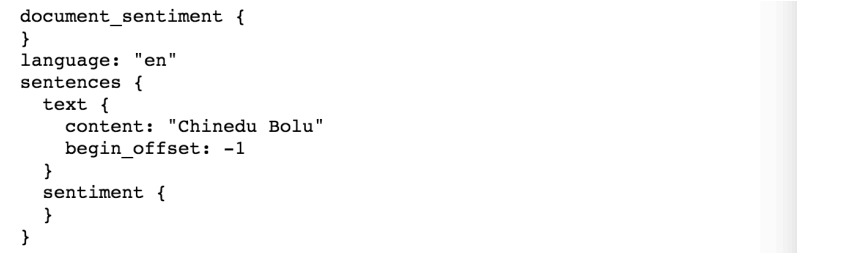


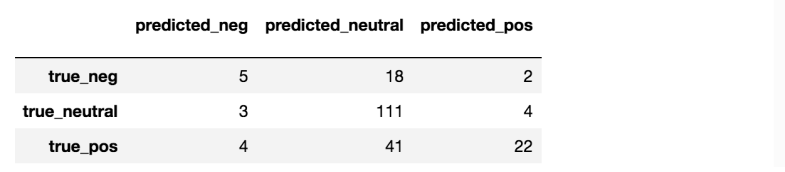
**8. Team8\_Kroger**



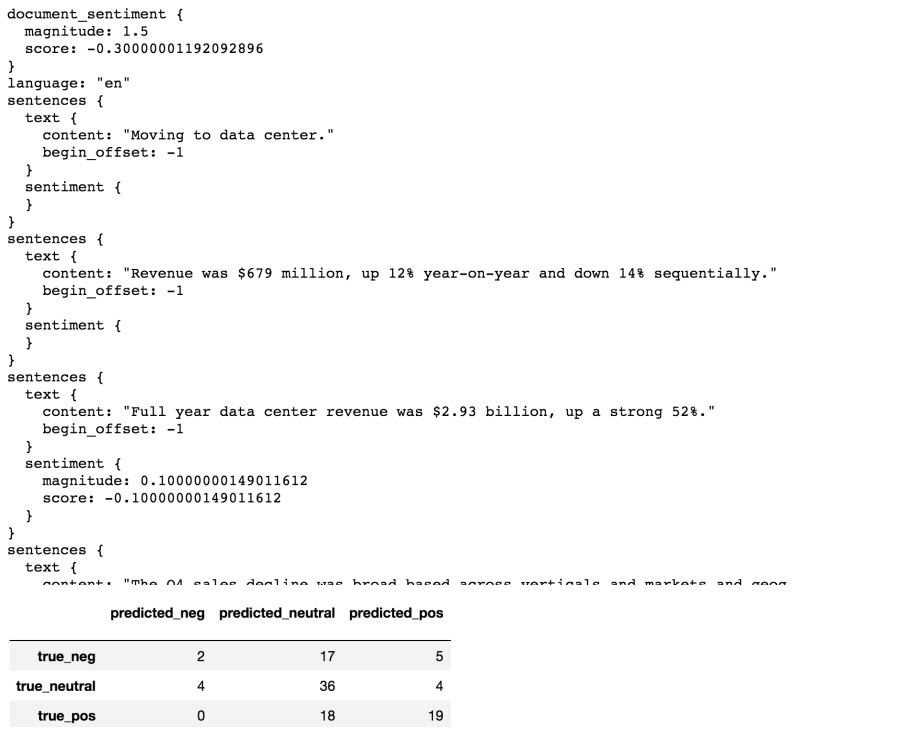


**9. Team9\_GoldmanSachs**

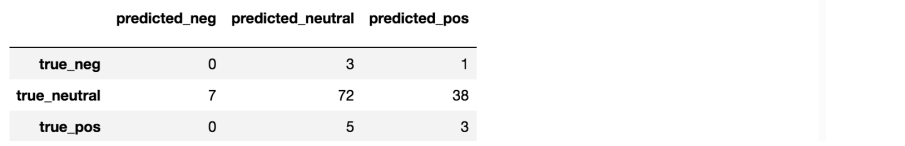
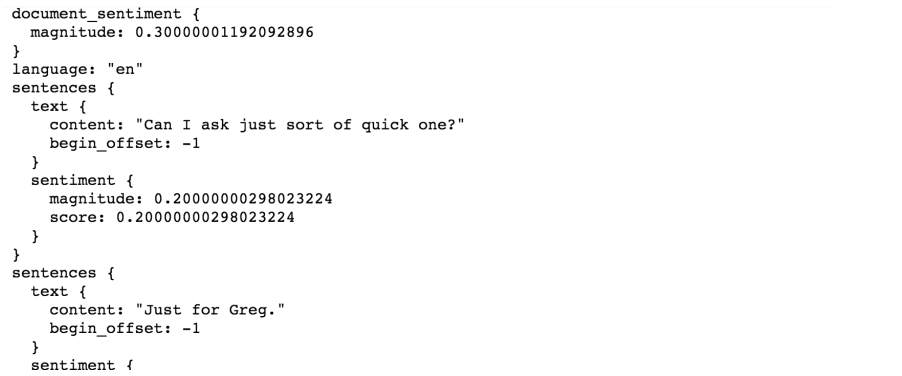




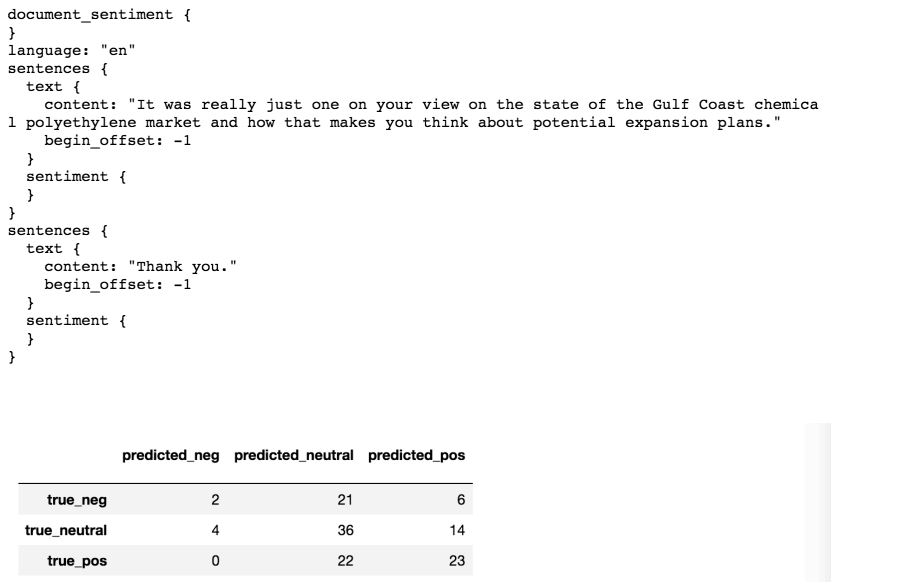
**10.Team10\_NVIDIA**

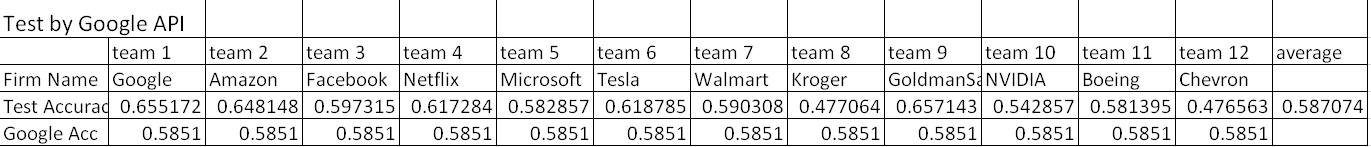


**11.Team11\_Boeing**



**12.Team12\_Chevron**





We can see from the table above, the average test accuracy is 0.5870 which is nearly to the original accuracy generated by Google API. So we can draw a conclusion that the Google API is steady for these datasets, and the accuracy is about 60% to give out the right sentiment labels with our mannually labeled ones. Because we use three kinds of labels to analyze, the average accuracy is 1/3, but the Google API improve the accuracy by 100%, so this is a good model. However, even human being labeled the pargragh, it is quite possible to make a mistake or change the standard when we make the labels. Besides, different people have different standard, but the Google API utilize the big data to anslyze the data in a more commen direction. So this is a good output for us to analyze the sentences and pragraghs. Certaintly, Goog API should continue to train more data to improve the accuracy from 60% to a hihger value.

Experiment 4:

For TPOT Automated Machine Learning tool, we put the pre-processed labeled texts data from 12 teams into the process. After finishing of the search, TPOT gives a model with score 0.6650717703349283. The best pipline:

Best pipeline: ExtraTreesClassifier(BernoulliNB(Normalizer(input\_matrix, norm=l2), alpha=100.0, fit\_prior=True), bootstrap=False, criterion=entropy, max\_features=0.8500000000000001, min\_samples\_leaf=16, min\_samples\_split=15, n\_estimators=100)

Using this model, we got the accuracy 0.6794 for test data set.

