Sentimental Analysis on Twitters by Using Neural Networks Based on Word2Vec Feature

Introduction

Sentimental analysis has long been regarded as an important part in natural language processing research area. From online rating system to fake news detection system, sentimental analysis help people to handle massive amount of data, to classifying and labelling nature language sentences.

Nowadays, social media such as twitter generate huge amount of data every days, which classic machine learning will face convergence when handling such amount of data. However, neural networks can tackle this problem and improving it accuracy with training data arising. In addition, a neural network can be developed by feeding more development data based on origin model, but classic machine learning algorithm may have retrain whole dataset to update parameters which cost times and computational ability.

In this work, we will try to using three different structure of neural network to build classifier to distinguish twitter's label between negative, neutral or positive. The main efforts in this work can be listed below:

1、Write code to pre-processing raw twitter data.

A、 Using regular expression to remove emotion and emoji, which represent as non-alphabetic characters .

B、 Replace URL links and user mentioned names with certain tag "usrurl" and "usrmnt" respectively, and remove hash tag for tags.

C、 Spell checking and elongated word correction.

2、Represent data as Word2Prob, Bigram2Prob and Word2Vector respectively

3、Write code to build a deep plain neural network and a many-to-one LSTM recurrent neural network based on Keras. LSTM has achieved much significant accuracy than other two classifier.

In this essay, the data processing detail will be illustrated, followed by an introduction of neural network structures that be implemented in this work. Evaluation of predicted result will be present before the conclusion. Furthermore, an appendix will be attached at the end of essay, to explain the detail of how to run the code.

1. Pre-processing

The pre-processing can be conclude in to three parts, which are tokenization, word replacement, and character filtering, and words spelling checking.

1.1 Tokenization

Compared with last assignment we had done, tokenize twitter would be more complex than previous work due to the special format in twitter, for instance hash tags , emotions and emoji characters. We change the strategy little bit, and we tokenize sentence by using following regular expression:

```
# reg for tokenize
regex_reg = [
    emoticons_reg,
    url_reg,                                # URLs
    r'<[^>]+>',                             # HTML tags
    r'(?:@[\w_]+)',                         # @-mentions
    r'(?:\#+[\w_]+[\w\'_\-]*[\w_]+)',       # hash-tags

    r"(?:[a-z][a-z'\-_]+[a-z])",            # words with - and '
    r'(?:[\w_]+)',                          # other words
    r'(?:\S)'                               # anything else
]
```

Figure 1: Regular expression for tokenization

With regular expression upon, we now not only tokenized data, but also divided it in different group. So, now we can process different group with different strategies.

1.2 Word Replacement and character filtering

Tweets may include user's mention which start with character "@" and URL link in the context. From sentimental analysis perspective, those token normally not contributed any meaning in positive or negative dimensions. So in order to reduce our vocabulary dictionary size, we can simply replace those group with "**usrurl**" and "**usrmnt**".

Technically, the emotion characters and emoji symbol may represent strong personal perspective of context. So we add it into vocabulary when we extracted Word2Index features. However, we didn't find an efficient way to transfer those symbol into context, and Word2Vector feature not support those symbol, so we have to remove it, when we extract Word2Vector features.

1.3 Words spelling checking

People who post on twitter not usually check their grammar and word spelling, even famous people like Donald Trump, president of America, always mis-spelling when post tweets. On the other hand, people who use social media like using elongated words to press their emotion, such as "amaaaaaaaaziiiiiiiiiiing", should be represented as "amazing". Under this situation, we took two steps to process those words.

First, we use regular expression to match the string who have character repeat more than two times, which English normally would not have this pattern. So we replace those repeat pattern and only remain one substring. For example, "amaaaaaaziiiiiiiing" will be turned to "amaaziing", which still mis-spelling word.

Second, now we can tackle the mis-spelling problem by using python package **textBlob** which can help us check our words and correct it spelling. It should turn "amaaziing" into "amazing". However, there still some abbreviation which we cannot handle, so in the further step we need a corpus that includes such sentence, to express it.

2. Feature Extraction

In this work, we consider using unigram (token) as baseline features, and further we consider the relationship between words, so that comes to bigram. In this stage, we represent words as a normalized probabilities in each class, for example, word I has been represent as:

$$I = [\frac{P(I \mid negative)}{sum(P)}, \frac{P(I \mid nuetral)}{sum(P)}, \frac{P(I \mid positive)}{sum(P)}]$$

Where $sum(P) = P(\text{I}|negative) + P(I \mid neutral) + P(I \mid positive)$. So, out training set can be organized as 3*n matrix, where n as n samples. However, in order to training our model in a parallel way, we have to re-organize it into a **1*(3n)** shape to fit vectorising calculation in numpy.

In the unigram representation, we have a significant drawback is that we did not considering the relationship between words. This brings a misunderstanding for some cases, for example “not” and “bad” may usually has some negative meaning, however when it come together, “not bag” is definitely positive collocation. So we use bigrams to replace unigrams, with same representation.

Another way to represent data is by using pre-trained Word2Vector set. In this work, we using **Glove.twitter.27B.50.** Who pre-trained from 27 billion tweets and represent each word as 50 dimension vector. For example “Languege” in word2Vecor is looks like:

[-0.451221,-0.4577887, ……, 0.878444] -> 50 dimension vector.

3. Neural Networks

In this work, we have implement two type of neural networks: full connected deep neural network and a many-to-one recurrent neural network. The skeleton code of this two neural networks is provided by Anderw Ng from Coursera.com. I have implemented my own code and my own hyper-parameters on this two neural networks, and modify it to suite our training set.

Full Connected Deep Neural Networks

Full connected deep neural networks, also called plain neural networks, it a basic structure of neural networks. In the multi-class classification problem, for example we have n class in training set, the output layer should have n sigmoid neuron that can output an n dimension vector to represent the probability of the samples in each class. The highest value indicate the most likely class the sample belong, the theory can be illustrated as below:

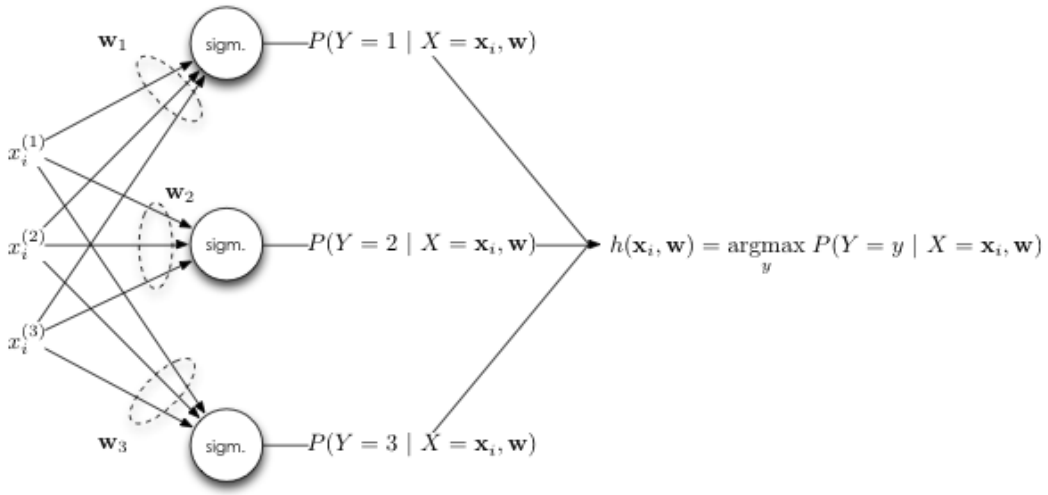

Figure 2：Multi-class classification problem in neural networks.
From braindolhansky.com [Accessed 2018/03/17]

As we have introduced in last section, our input data has 1*3n dimensions, so our input layer should have 1*3n neuron. As twitter has 140 word count limitation, we can simply set the n as 140. The output layer should be set as 420 neuron layer.

To accelerate the forward propagation and backward propagation during training, the activation function of hidden layer has been set as ReLu (rectified linear unit) function.

Long Short-Term Memory Network

In a long short-term memory network, it has three “gates” and two activation units, based on this structure we could previous memory (value) to decided current state and then update the state for next LSTM cell update. The memory will update through this LSTM chain, and nearest cell has majority influence on it neighbours.

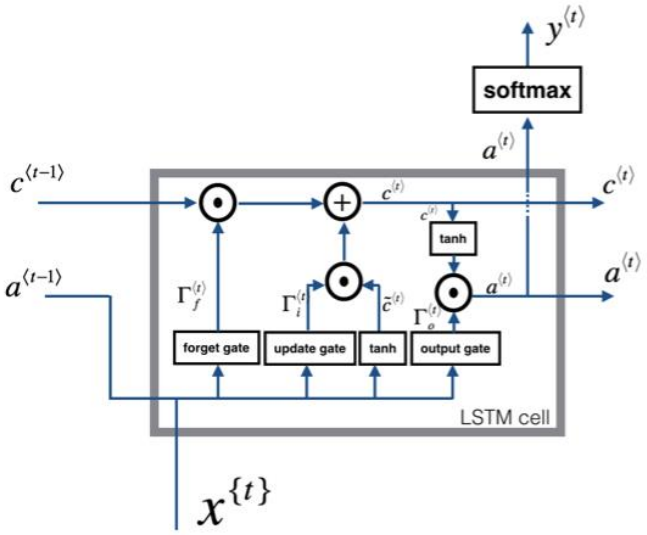

Figure 3: Long Short-Term Memory Cell

From Coursera: Sequence Model week two. [Accessed 2018/3/1]

In this network, we use word2vector feature as a sequence input. The index of word will be input into a pre-trained embedding layer, to transfer to a 50 dimensions vector. After first LSTM layer a dropout layer is been used to regularize the network. The output layer only has one neuron, after last layer’s last dropout cell. A soft-max cell to generate final predict result. The structure of network is showing below:

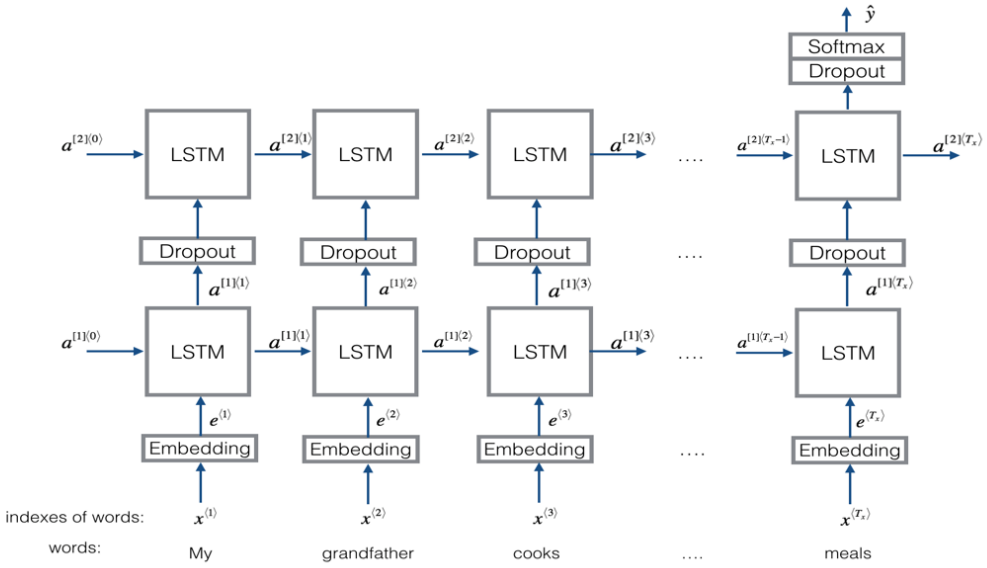

Figure 4：2-layers LSTM Sequnce Many-to-One Classiifier

From Coursera: Sequence Model week two. [Accessed 2018/3/1]

Evaluation

Training results of each classifier are performed quite good accuracy. However, it seems to meet an overfitting problem during test, the accuracy in test set are significant lower than training accuracy, table below shows the classify result under cost threshold = 0.001:

Classifier-threshold	Value Type	Training	Test1	Test2	Test3
Uni-0.001	Accuracy	0.8283	0.5706	0.5940	0.5611
Uni-0.001	F1-value		0.445	0.461	0.435
Bi-0.001	Accuracy	0.9946	0.5239	0.5278	0.5103
Bi-0.001	F1-value		0.337	0.378	0.318
Uni-0.01	Accuracy	0.8283	0.5707	0.5974	0.5612
Uni-0.01	F1-value		0.445	0.461	0.435
Bi-0.01	Accuracy	0.9945	0.5217	0.5273	0.5115
Bi-0.01	F1-value		0.334	0.370	0.318
2-LSTM	Accuracy	0.8421	0.6256	0.6503	0.5822
2-LSTM	F1-value		0.602	0.642	0.570

Figure 5: F1-value and Accuracy through all Classifier

From this table, we can see that 2-LSTM has minimum variance and least bias through all classifier. Now we can look

Test1	positive	negative	neutral	Test2	positive	negative	neutral
positive	0.700	0.046	0.254	positive	0.779	0.023	0.198
negative	0.220	0.493	0.288	negative	0.216	0.514	0.270
neutral	0.284	0.085	0.630	neutral	0.374	0.064	0.562
Test3	positive	negative	neutral				
positive	0.725	0.046	0.230				
negative	0.202	0.400	0.398				
neutral	0.337	0.085	0.578				

Figure 6: Confusion Matrix for 2-LSTM

Base on this table, the boarder between positive and negative, neutral and negative have already quite clear. However, this classifier still little bit hard to label the negative samples. The overfitting problems happens in some boarders, may need more data to be distinguished.

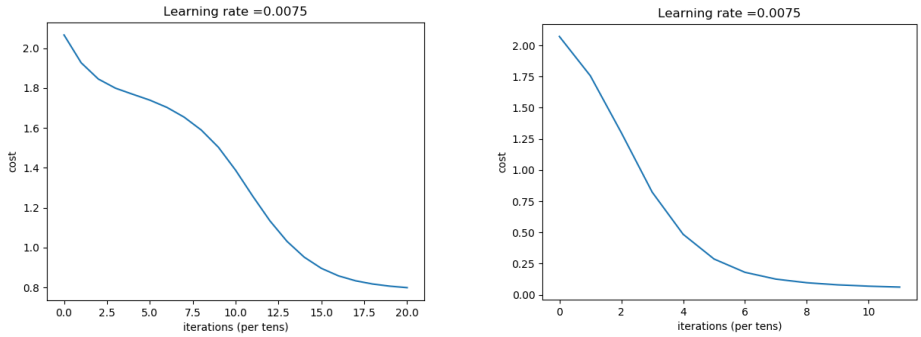

Figure 7: Cost Descending

Conclusion

In this work we build two structure of neural networks to process three type of features. The sequence model classifier seem to be the best solution for sentimental analysis in all those classifier. There still some improvement of this work can be done. First, the emoji and emotion characters always represents strong emotion, we need build a vocabulary to translate it. The hash tags always shows the core means of a twitter, in the future work, we may need to treat it as a new dimension of features. In conclusion, neural networks have great potential on solving sentimental analysis problems.

Program Appendix

Q: How to run this code?

A: This project is developed based on **python 3.6.4.** Several package has been used to tackle certain task: Here is a table listing the package need to be installed before running code.

pickle	Can be installed by pip or Anaconda
numpy	Can be installed by pip or Anaconda
keras	A learning framework can be easily installed in Anaconda, tensoflow required.
matplotlib	Can be installed by pip or Anaconda
nltk	Need bigrams, stopwords etc
textblob	Can be install by pip

After installed all this package, a pre-trained word2Vect **Glove.twitter.27B.50.** required to be put in to **semeval-tweets** folder. Then you can run this code by type command in to terminal: **python classification.py.** The process of pre-processing and model training may takes several minutes to be done.

Q: I want to use trained model to evaluate

A: The pre-processed data set and trained model are not submitted through tabula, due to the limitation of submission requirement. The whole project has been uploaded on figShare, so that you can access all codes, data set and trained model.

https://figshare.com/articles/cache_rar/6004094

All pre-processed data set and trained model, should be placed under **cache** folder

Q: Where can I download Glove.twitter.27B.50.?

A: Glove.twitter.27B.50. is a public dataset which can be accessed from Stanford University: https://nlp.stanford.edu/projects/glove/

http://nlp.stanford.edu/data/glove.twitter.27B.zip