JSC270 A4 Report

Shi Tang, Yongzhen Huang, Jianhong Guo

April 9, 2021

1 Breakdown

- Analysis (coding portion): Yongzhen, Jianhong, Shi
- Presentation: Yongzhen, Jianhong, Shi
- Report: Yongzhen, Jianhong, Shi

2 Sentiment Analysis with a Common Twitter Dataset

2.1 A

The proportion of the three classes is unbalanced. As calculated in the notebook, class 0 (negative) has a proportion of 0.374159, class 1 (neutral) has a proportion of 0.187407, and class 2 (positive) has a proportion of 0.438434. By comparing these values, the neutral class is much smaller than the other two.

2.2 B

For each observation, we converted the tweet from a single string of running text into a list of individuals tokens and split it on white-space by applying *WhitespaceTokenizer().tokenize*. Please refer to the code.

2.3 C

To remove any URLs tokens from each of the observations, we used white-space to replace the any sequence that satisfies the regular expression pattern: "$(http|www)nS+$". Please refer to the code.

2.4 D

In this part, we only keep words, white-space, and removed all punctuation $(,.?!;)$ and special characters $(@,,+,,=,\$)$. Please refer to the code.
There are some scenarios when people might want to keep some forms of punctuation, for example, if people want to count the length of a sentence, they may want to keep all the commas. Also, if we want know the emotion of a sentence, we may want to keep "!" since exclamation mark is used to end a sentence expressing strong emotion or may be used to close questions that are meant to convey extreme emotion (What were you thinking!).

2.5 E

We want to convert similar word forms into identical tokens, to do so, we applied the Porter stemmer to stem the tokens. Please refer to the code.

2.6 F

Removed the top 100 stopwords by using the English stopwords list from nltk. Please refer to the code.

2.7 G

After converting the lists of words into vectors of word counts using Scikit-learn's CountVectorizer, the vocabulary size is 55423.

2.8 H

Fitting a Naive Bayes model, we obtain:

- The train error is 0.21, and the test error is 0.32.
- The train accuracy is 0.79, and the test accuracy is 0.68.

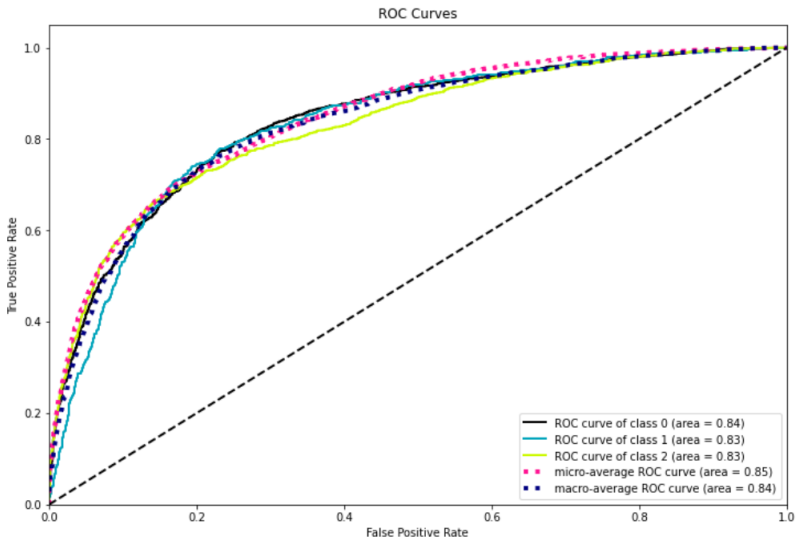
The most five probable words in each class:

- Sentiment 0: ['coronaviru' 'covid19' 'price' 'food' 'thi']
  Counts 0: [6703 4862 4332 3622 3206]
- Sentiment 1: ['coronaviru' 'covid19' 'store' 'supermarket' 'price']
  Counts 1: [3792 2751 1581 1435 1361]

- Sentiment 2: ['coronaviru' 'covid19' 'store' 'thi' 'price']
  Counts 2: [7466 6001 3895 3770 3322]

2.9 I

It would be appropriate to fit an ROC curve because the count of each class is balanced. In a in-balanced class distribution, the ROC curve does not give much information since the AUC will be very high. In this case, the three classes (0, 1, 2) are $44\%, 37\%, 19\%$ of total data, respectively. So the classfication does not favor any particular class and thus ROC curve is applicable in this case. The graph is below:



2.10 J

Applying TF-IDF vectors by using Scikitlearn's TfidfVectorizer() transformer, we obtain the following results:

- The train accuracy is 0.72 and test accuracy is 0.64.
- The accuracy of using TF-IDF vectors are smaller than the accuracy using count vectors, but the differences are not too much.

2.11 K

Redo parts E-H using TF-IDF vectors instead of count vectors, using WordNet lemmatizer offered by nltk, we obtain the following results:

- The train accuracy is 0.73 and test accuracy is 0.64.
- The accuracy of lemmatization is quite close to the accuracy with stemming. The train accuracy of lemmatization is 0.01 higher than the train accuracy of stemming.

2.12 Bonus

Naive Bayes model is generative since Naive Bayes model is according to the probability of p(x, y) where x is the inputs and y is label to find the most likely label y by using Bayes rule to calculate $p(y|x)$.

3 Next Word Prediction

3.1 Problem description and motivation

This assignment uses data from Twitter API to predict the next word. Word prediction is one of the fundamental functions of NLP and has many applications! People might use it daily when they write texts or emails without realizing it. The word prediction system speeds up communication rate by 58.6.%[1] and helps people with severe spelling problems.

Even though language prediction is one of the fundamental tasks in NLP, people still facing some challenges when building language models. Words have different meanings depending on the ambiguity of the lexical, syntactic, and semantic levels, and people can express the same idea in different terms.

As an important social media platform, Twitter plays a significant role in people's daily life. Our group decided to build a language prediction model using Twitter data and to explore underlying language patterns. According to research, there are generally some commonly used models for word prediction, for instance, the N-grams model, Long short-term memory (LSTM), SNoW Architecture [2] and Markov models. In this assignment, we decided to explore a more advanced approach, using a neural language model, which is to use LSTM. In brief, the LSTM model uses Deep learning with a network of artificial "cells" that manage memory, making them better suited for text prediction than traditional neural networks and other models.

3.2 Describe the data

For data extraction, we extracted tweets with the parameter "lockdown" and filter out all the retweets to avoid duplication. We decided to extract tweets that contain "lockdown" because Ontario entered the third COVID-19 lockdown, and we want to build a language prediction model to see how people react to the announcement of lockdown.

There is a total of 100000 tweets contain the keyword "lockdown" which is used for analysis. There are some limitations of the data by the nature of tweets. Most of the extracted tweets are very casual containing buzzwords and shortcuts which increases the difficulty to obtain the underlying language patterns and makes the model harder to generalized. Unfortunately, no other similar works of building language prediction models using tweets are found.

3.3 Exploratory data analysis

When we first extracted the tweets, there are noises in the original text which contains many special characters, and the same word may have different tenses. In order to better predict the word, pre-processing step is required.

In the pre-processing step, words are tokenized, URLs, usernames and special characters are removed. All words are changed to lower cases, and lemmatization is applied to the tokens. The pre-processing examples are shown below:

	tweets	tokens
0	Covid: Croquet clubs 'inundated' during lockdo...	[covid, croquet, club, 'inundated', during, lo...
1	Lockdown Madness In Cebu City 4/4/21 https://t...	[lockdown, madness, in, cebu, city, 4421, via]
2	@dhanushkraja Everyone is talking about Lockdo...	[everyone, is, talking, about, lockdown, from,...
3	'The zebras loved lockdown but the goats misse...	['the, zebra, loved, lockdown, but, the, goat,...
4	#FMCG companies getting ready in anticipation ...	[fmcg, company, getting, ready, in, anticipati...
5	Corona Report from Medical officer/ DM has not...	[corona, report, from, medical, officer, dm, h...
6	#Lockdown in #bangladesh \nFor 1 week. https:/...	[lockdown, in, bangladesh, for, 1, week]
7	I am rewatchint the Amy's Baking Company episo...	[i, am, rewatchint, the, amy's, baking, compan...
8	When people & Business owners r ready to p...	[when, people, amp, business, owner, r, ready,...
9	🚨🚨🚨MINGYU LOCKDOWN TOMORROW🚨🚨🚨🚨🚨🚨🚨	[mingyu, lockdown, tomorrow]

In order to use the LSTM model, more data analysis is needed. First, tweets that have a length smaller than 6 are removed since those tweets contain too little information and those are too short to be predicted.

Then, after converting tweets into sentences, we converted sentences into sequences. Besides, each sequence is sliced into sequences with a length of 16 to increase the input sizes. To illustrate, for a sequence A with a length of 18, we can generate 3 sub-sequences A[0:16], A[1:17] and A[2:18]. This process is shown as below:

Tweets	text to sequence	sequence	sequences with length 16
[['the', 'way', 'i', 'havent','touched', '95', 'of', 'my', 'belonging', 'since', 'the', 'lockdown', 'i', 'feel', 'this', 'time', 'is', 'better'],...]	{'actually': 33, 'after': 23, 'also': 49, "amp": 87, "and": 65 ... }	[[19, 20, 21, 22, 23, 1, 24, 25, 26, 6, 3, 1, 27, 28, 29, 2, 30, 31], ...]	[[19, 20, 21, 22, 23, 1, 24, 25, 26, 6, 3, 1, 27, 28, 29, 2], [20, 21, 22, 23, 1, 24, 25, 26, 6, 3, 1, 27, 28, 29, 2,30], [21, 22, 23, 1, 24, 25, 26, 6, 3, 1, 27, 28, 29, 2,30,31]...]

After all the above steps, we obtained a vocabulary size of 53,916, a total of 1,254,048 sequences with a length of 16. Each of the sequences is separated into X and y while X has the first 15 words, and y contains the last word as the label. Finally, X and y are separated into training and testing data with train size 0.8.

3.4 Describe machine learning model

The machine learning model used is based on LSTM implemented using Keras. Keras is a high-level neural network library.

As shown in second figure below, the model contains the following layers

1. Embedding layer
2. Bidirectional LSTM
3. LSTM
4. Dense layer with ReLU activation function
5. Dense layer with Softmax activation function

Together, we use the sparse categorical cross-entropy as the loss function and Adam as the optimizer.

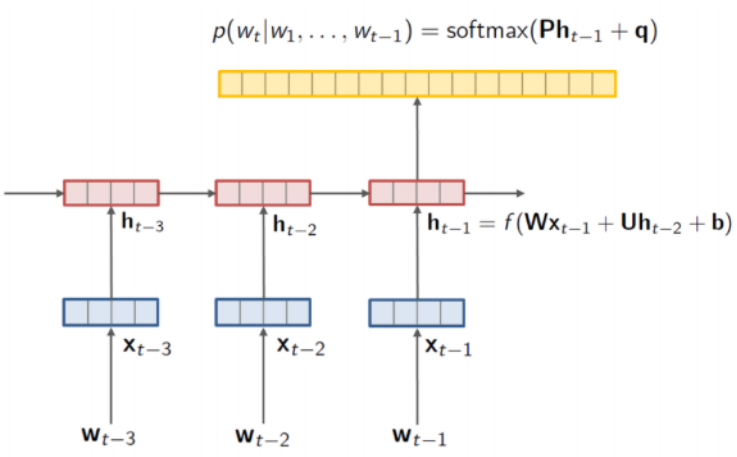
Using below figure as a simplified example, the model takes sequences of 15 integers as input; the mapping to the words in the vocabulary. The sequences are passed to an embedding layer which transforms the words into a space the model can better understand. The embedded sequences are then passed into RNN-type networks (LSTM in our case). Compared to other RNN architectures, LSTM has forget gates which avoids vanishing gradients and hence has better stability. Furthermore, the first LSTM we use is bidirectional exposing the forward and backward phrase structure to the model (without exposing the actual label i.e. information leakage).

After the LSTMs, we have a couple of fully-connected layer. In the last layer, we have softmax activation function which calculates the probability of each word in the vocabulary being the next word in the phrase (the 16th word). Hence the output dimension is the size of the vocabulary. The word with highest probability is selected as the predicted word. Finally, the prediction is tested against the label to calculate categorical cross-entropy loss for training and optimization of the model. The learning rate we use is at least 0.001, it starts higher and will reduce if accuracy is near a plateau.

One thing worth mentioning is the use of categorical cross-entropy versus sparse categorical cross-entropy. They perform the same function but the latter is good for very large models such as ours with large vocabulary size (i.e. categories). Regular categorical would use $N \times M$ numbers where N is number of samples and

M is the category or vocabulary size. However, only N of these are non-zeros. Therefore, it is much more memory-friendly to use the sparse version to avoid crashes due to insufficient memory. Indeed, what we used is sparse version.

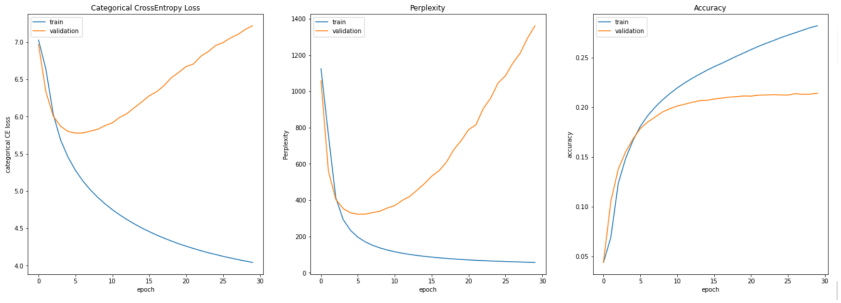
This model is an appropriate choice since it can learn from temporal structure (past or future context). Also, it can be used for generalization. However, this model also has some weaknesses, for example, it is expensive to train and test, and requires a large amount of data. This is a supervised model because the next words are used as labels. The model's performance is evaluated using perplexity which will be discussed in the conclusion. Unfortunately, no other similar works were found, so there are no baseline models to compare.




```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 15, 20)            1078320
_________________________________________________________________
bidirectional (Bidirectional (None, 15, 200)           96800
_________________________________________________________________
lstm_1 (LSTM)                (None, 100)               120400
_________________________________________________________________
dense (Dense)                (None, 100)               10100
_________________________________________________________________
dense_1 (Dense)              (None, 53916)             5445516
=================================================================
Total params: 6,751,136
Trainable params: 6,751,136
Non-trainable params: 0
_________________________________________________________________
```

3.5 Result and Conclusions

The accuracy of the model is 21%. The parameters of the model have a meaningful interpretation, such as the number of epochs. As it shown in the perplexity graph, the model is over-fitting after 10 epochs, and the accuracy is less affected by epochs after 10, so choosing an appropriate epoch can reduce the over-fitting problem. Also, we choose bidirectional instead of directional in our model since bidirectional gives a higher accuracy.



This approach has some limitations compared to exiting approaches, for instance, tweets are usually informal, so it's harder to find underlying structure (e.g. grammar). Also, due to the limitation of data quantity, the accuracy is not good enough. If we have more time and could collect more data, choosing the data from few selected people or from some organizations is helpful since this kind of data has more patterns. Moreover, applying techniques such as drop-out will help the model to perform better too.

4 References

[1] Keith Trnka, John McCaw, Debra Yarrington. WORD PREDICTION AND COMMUNICATION RATE IN AAC. April 16-18 2018. Proceedings of the fourth IASTED international conference.

[2] Yair Even-Zoharr, Dan Roth. A Classification Approach to Word Prediction. https://www.aclweb.org/anthology/A00-2017.pdf