Jake Cross

CS461

6 March 2022

Program 3 Report

1. **Data Preparation**

I drop full\_text, hashtags, and year columns at first. Full\_text and hashtags does not work for my model. Year is not a good feature. However, we need to treat hashtags and full\_text for machine learning. For hashtags, I got top 500 hashtags by using numpy.unique stuff. I add each of the tags and tag ‘Other’ (not in top 500 hashtag) as column of panda. If their tweets involve one of the top 500 hashtags, it assigns 1 to the column. Otherwise, add 0 and add 1 to column ‘Other’.

Table

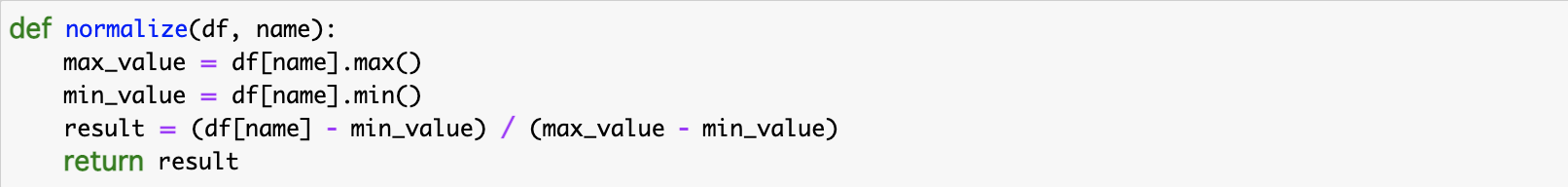
Description automatically generated

For full\_text, remove hashtags, punctuations, RT keyword, and remove stop\_words after all cases changed to lower cases. Then get the length of modified strings and assign it to column ‘length’.

Graphical user interface, text

Description automatically generated

Except ‘party\_id’, all integer values are normalized by min-max normalization.



All values are converted to ‘float32’ datatype, so we do not conflict on data types.

For categorical\_crossentropy, I applied LabelEncoder() to party\_id so that all strings are converted to integers. After that, I did one-hot encoding.



1. **Network Configuration**

Input shape is 504. Output shape is 2. There is one hidden layer with relu activation function that returns Max(0, x).

The number of neurons is 512-256-2. I add two dropout layers so that it can learn properly. I did one-hot encoding to the label so that I can use categorical\_crossentropy.

1. **Validation Strategy**

****

I applied the above code. In order to get training and testing dataset, I used train\_test\_split. 75% of data is randomly selected for training and 25% for testing. For each epoch, I applied validation\_data function to get validation loss and accuracy which are used for callbacks. I used three callbacks which are ModelCheckpoint (save each epoch to the file), ReduceLROnPlateau (reduce learning rate for gradient descent), and EarlyStopping (stop learning if validation loss or accuracy would not be updated). We can ES works since it quit fitting before all epochs taken place. These callbacks prevent my model from overfitting. I also reduce batch sizes, so it takes more time for training, but the risk becomes low.

1. **Results**

Its accuracy is 72%.

Graphical user interface, text, application

Description automatically generatedIt is not so bad, but the accuracy did not change in each epoch so much.

Graphical user interface, text, application, email

Description automatically generatedI tried the following things.

* Add more dense layers
* Add dropout layers
* Change activation function relu to tanh
* Change activation function softmax to sigmoid
* Change optimizer rmprop to adam

All of these did not affect so much. I will discuss my assumption is the next section.

1. **Comments**

I am taking python deep learning course and worked similar project a few weeks ago. In the project, I convert all text to integers so that I can use Embedding and LSTM layers. I just used text to predict the class of tweets and get 63% accuracy while I get 71% accuracy on this project. I believe my model is not so bad since 71% is not low and the model’s prediction works well. However, in the training, the accuracy does not change so much. It looks like overfitting even though we have plenty data. As I mentioned, I tried to several things, but it does not update the result. I have never worked on the model has more than 100 inputs, so I honestly could not imagine what is going on in the model. One thing I can guess is probably my dataset does not have good features to show high accuracy. It might not be good to predict someone’s party from tweet length, the number of retweets and favorites, and hashtags or one of them makes it complex.

1. **References**

* [Apply np.unique to get high frequency values](https://stackoverflow.com/questions/48784908/numpy-unique-sort-based-on-counts)
* [Simple sequential model](https://www.kdnuggets.com/2018/06/basic-keras-neural-network-sequential-model.html)

1. **Possible changes**

* Temperature value should be higher: I set very large number to the temperature, but it should be much higher to get more accurate result.
* More efficient way for random selections: I think numpy.random.select is a more simple and faster way than vectors of booleans, but there might be much better ways in Python.