Sentiment Analysis on Yelp Reviews

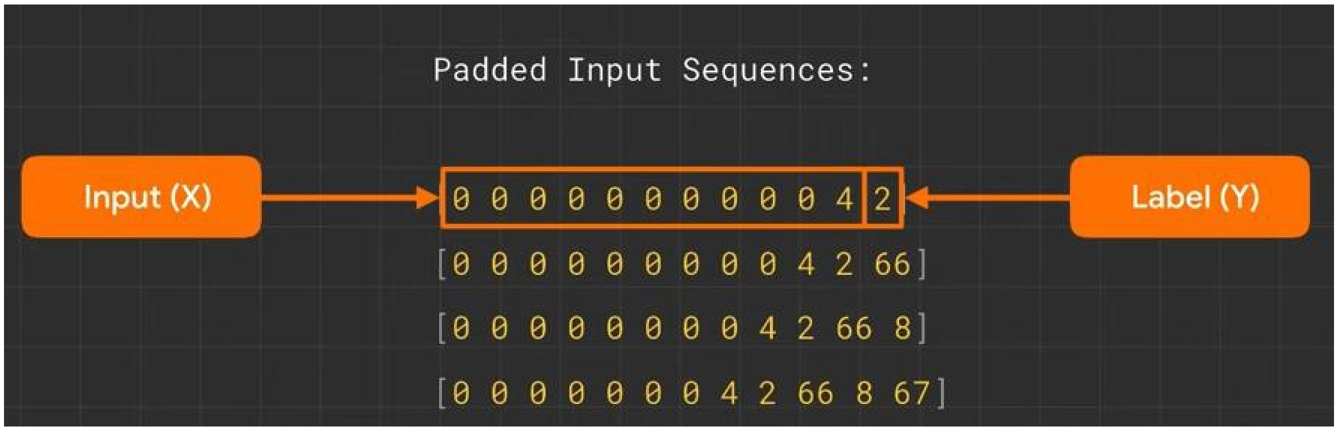
In this paper I will demonstrate the creation of a neural network using Natural Language Processing (NLP) and Tensorflow. NLP is a machine learning technique, also called deep learning, that develops a neural network capable of predicting user sentiment. In this paper I will be using written Yelp reviews on restaurants. My research question is as follows: how well will our network accurately predict new customer sentiment given our training data? How can we use the TensorFlow API for understanding customer sentiment?

For any aspiring restaurant, the above research question has tremendous value for future revenue streams. Since we can gather information related to the general sentiment of our customers, we can make better informed decisions about how to expand our business into new territory or change our business practices.

The goal of this analysis is not only to create a usable model that can predict customer sentiment on unseen data, but also show the mathematical patterns of sentiment so that we can achieve a quality evaluation of the training set and then make a recommendation to the appropriate stakeholders.

Identifying the proper neural network is an important step when it comes to machine learning, as many networks perform worse or better depending on the task at hand. Specifically for text classification regarding sentiment the proper machine learning topology is the Recurrent Neural Network (RNN).

RNN works well with sequential data and therefore has an advantage over other networks like CNN (Convolution Neural Network) which are commonly used for image recognition. Below I have included a picture representation of an RNN:



Essentially, RNN places an integer value for each word in a sentence. Then, using the longest phrase in the corpus as the maximum index for the list of integers representing each phrase, the shorter encoded phrases are "padded" with 0 for the delta between them. We pad 0s before each word in each phrase as well, and then create a list of each phrase with the subsequent word added as a new list (as shown above).

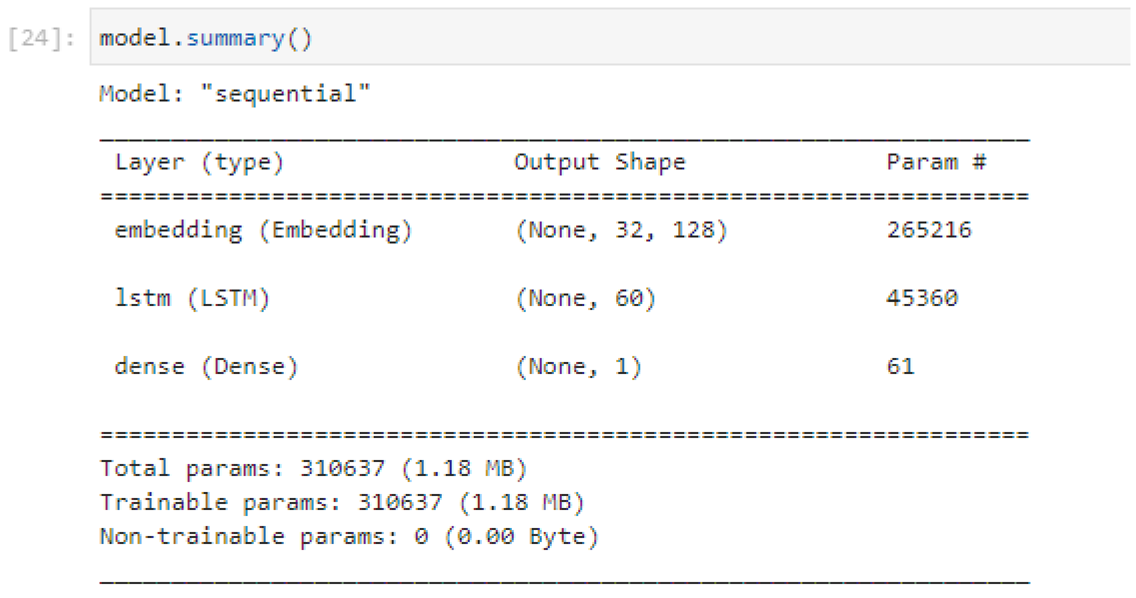
It is because of this feature that we can represent words sequentially as a matrix of vocabulary "tokens" and therefore use neural networks to predict an outcome mathematically. In the above picture, the padded input sentences are used as the 'regressors' while Y-hat is the label we want to predict.

Our next step is to perform EDA on our data; we look for missing or null values, change our sentiment column to a binary classification (1 for positive or 0 for negative), check for non-alphabetic characters and remove them. Our dataset is very clean as it has no null values and our sentiment columns are in the appropriate binary form. We then print the vocabulary size to the console and find our embedding length.

Embedding length refers to the longest tokenized vector from our input reviews (df_yelp[‘Reviews]) and each vector that is less than the maximum will be “padded” with zeros. I choose to pre pad which means we placed zeros in front of each tokenized sentences. Technically, the embedding size is the fourth root of the vocabulary size, which for us is 2072.

Padding is an import step because a neural network requires our input to be a vector of numbers (in our case, the vector is our tokenized reviews). Neural nets require that all inputs have the same input and shape, and therefore we use 0s to represent empty tokens to achieve this. We call the pad_sequences method on our data to get our padded inputs that we then feed into the model. Our data is from Yelp, which has a review column and a sentiment column with 1 for positive sentiment and 0 for negative sentiment. We prepared the data by performing the following:

1. Input the data into a data frame in the Python environment
2. Check for unusual characters and stop words
3. Clean the data by calling appropriate methods to check for missing or null data
4. Tokenize the reviews with tokens taken from the word_index
5. Split data into 80:20 training and testing data
6. Apply pad_sequences to training and test sets to achieve same shape and size
7. Fit the model with the training data



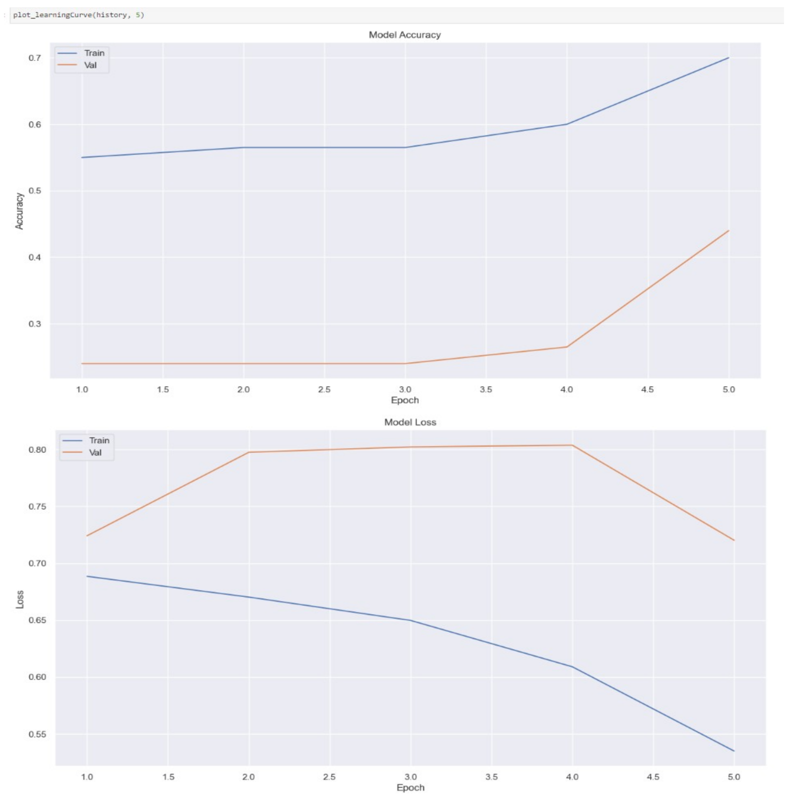
```
[24]: model.summary()

Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 32, 128)           265216

 lstm (LSTM)                 (None, 60)                45360

 dense (Dense)               (None, 1)                 61

=================================================================
Total params: 310637 (1.18 MB)
Trainable params: 310637 (1.18 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

We initialize our model with an embedding size of 128, which will eventually reduce to our dense layer with a single shape (output). We have 310,637 total parameters that were trained and weighted. As can be seen by our training data, we achieved a total accuracy of 0.74 over 10 epochs.

Let’s discuss some of the tuning done in this model and justifications for certain paraments in the keras model. Firstly, we have our embedding layer which reduces the information from our inputs into a tensor with shape 32, 128. 32 is 32 bits and 128 is our embedding size. This gets reduced into our next LSTM layer which outputs into a new embedding size of 60. Finally, our model output embeds into a single output. We train the neural networks in 10 epochs, which means this neural net back-propagates 10 times to re-weight the inputs. Our model accuracy is reflected as 0.745 after 10 epochs. Sigmoid activation is used for our dense layer since our expected output is a binary classifier (neural networks use logistic regression to predict). We use the EarlyStopping method to prevent the model from overfitting the data with a patience of 2 since 2 is very typical in this specific application of machine learning.

Using EarlyStopping can be a more holistic approach to model training over just changing the epochs, since the EarlyStopping method will track the improve in accuracy and stop when accuracy is not improving, it can provide us with a better model than just purely experimenting with the number of epochs.

Our model accuracy can be shown below, which demonstrates that our model's improvement is negligible after just 2 epochs, indicating that while our training data has good accuracy, it decreases quickly after 2 epochs. This is backed up further by our model loss which increases after 2 epochs.



```
plot_learningCurve(history, 5)
```

In order for our model to generalize well, it can not overfit the data. In our case, a possible solution to this problem is probably an increase in the amount of training data. Our training and test splits were 80:20 with 1000 total observations and a vocab size of just 2072.

Since we implemented a Recurrent Neural Network to gauge customer sentiment, with only a few layers in the network, there are some possible improvements that could be made on this network architecture. Increasing the epochs had no effect on model accuracy, and it did not perform exceptionally well on the testing data. It is possible that we could increase our hidden layers, or make adjustments to the activation functions that we use to activate neurons. When it comes to machine learning, a lot of generalization problems come down to having very good testing data, a lot of it, and tweaking models through trial and error.

I did perform a little bit of tweaking to the model, initially starting with 7 epochs, then 10, then five based on the model's performance. Based on my experience with this specific data, I would recommend that this neural network, since it does not generalize well on testing data, have the training data increased so that our neural net has more vectors to work with.

One possible solution would be to aggregate more reviews from other sources or increase the number of features by generating more data from Yelp’s database system. More data means our neural network has more information of which to train and reweight the biases between neurons, giving the network a better fundamental starting point to generalize well on testing data. The entire point of using machine learning is to make better informed decisions, which means our inputs need to generalize well on new data.