Sentiment Analysis of Movie Reviews Obtained from IMDB

Carson Hom, Jai Lunkad
[redacted]

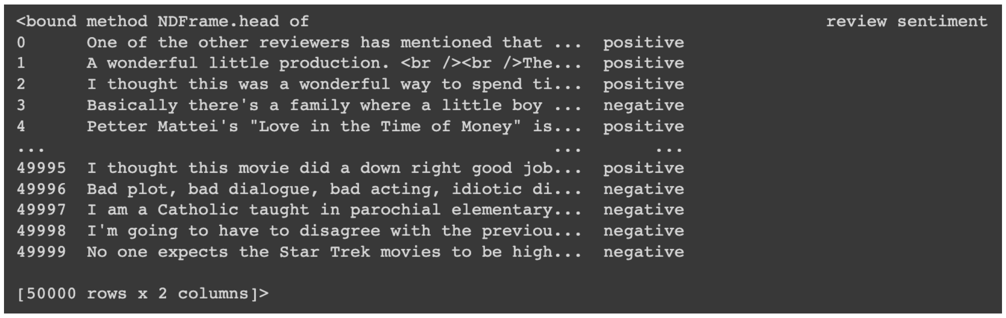
1. Introduction

The goal of this project was to develop a machine learning model which would be capable of determining the sentiment of movie reviews posted online by film critics. The dataset that we chose to build our model on contains 50,000 polarizing movie reviews from the popular film discussion website, IMDB. The dataset itself contains the plaintext reviews as well as the comments sentiment classification designating the review as either positive or negative.

Based on the characteristics of the dataset and the problem, the approach that we took involved using a Long Short Term Memory (LSTM) neural network. We settled on using an LSTM neural network over a feed forward neural network because of their ability to process entire sequences of data rather than just single points. In our case, the model needed to be capable of recognizing sentiment of entire plain text film reviews. In this paper we will discuss the steps taken to preprocess the data, build and train the model, analysis of the model and finally further applications as well as extended use of the results.

2. Data Preprocessing

The first step in data preprocessing was to identify the target feature in the dataframe. For the purpose of our project, the target feature was the movie review's sentiment classification column.



```
<bound method NDFrame.head of                                                     review sentiment
0      One of the other reviewers has mentioned that ...  positive
1      A wonderful little production. <br /><br />The...  positive
2      I thought this was a wonderful way to spend ti...  positive
3      Basically there's a family where a little boy ...  negative
4      Petter Mattei's "Love in the Time of Money" is...  positive
...                                                  ...       ...
49995  I thought this movie did a down right good job...  positive
49996  Bad plot, bad dialogue, bad acting, idiotic di...  negative
49997  I am a Catholic taught in parochial elementary...  negative
49998  I'm going to have to disagree with the previou...  negative
49999  No one expects the Star Trek movies to be high...  negative

[50000 rows x 2 columns]>
```

Next, we split the entire dataset of 50,000 reviews into a training set and a testing set where the testing set accounted for 30% of the total samples. We agreed on this particular split in the data based on previous experience with the homework assignments and in class examples.

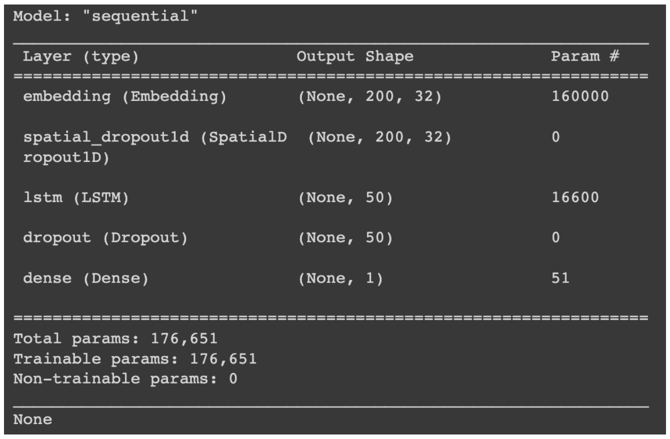
The next portion of the data preprocessing step was to convert the target feature from a categorical representation to a numerical representation. Since the two possible feature values are either ‘positive’ or ‘negative’, we chose to convert these categories to a binary representation where ‘0’ replaced ‘positive’ and ‘1’ replaced ‘negative’. This was accomplished using the factorize() function which performs this task automatically. We didn’t think that one hot encoding would be necessary since sentiment is a binary feature and also that performing a simple encoding wouldn’t introduce any patterns that did not already exist in the data.

After we had processed the sentiment feature, we also processed the review feature in order to simplify the step of training the model on the data. The first step involved breaking down each of the plain text reviews using a tokenizer. In tokenization, larger sentences are broken down into vectors of single words which makes it easier for a model to interpret the relationship between specific words. However, training our model on plain text words would add complexity and training time to the overall project. This is why we chose to use the fit_on_texts() method to create an association between words and an assigned number in order to create a sentence representation that would be easier for the model to understand. The association between words and their assigned number is stored in the tokenizer.word_index attribute where it can be accessed later. We replaced the words with their numerical representation using the text_to_sequence() method.

The last step in our data preprocessing portion of the project was padding out each of the reviews to be of equal length. We recognized that the varying lengths of reviews could pose a challenge to our model and so we took the step to padding out each of the reviews to a uniform length of 200 words.

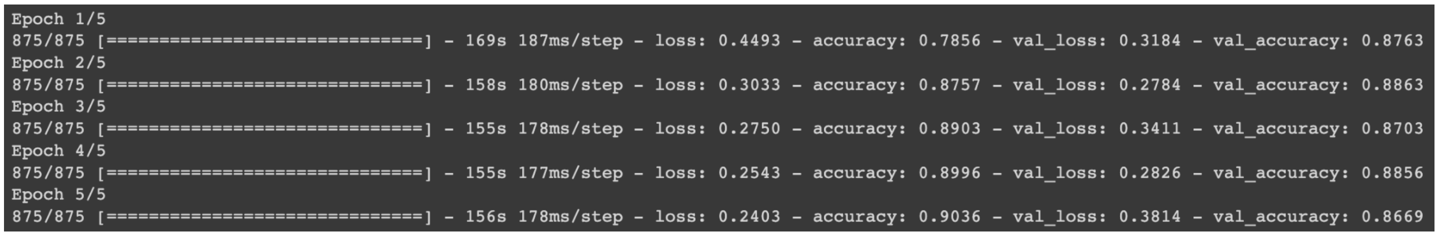
3. Building and Training the Model

In order to build a model that was capable of sentiment classification, we implemented LSTM layers in the machine learning model. The architecture for our model consists of an embedding layer, an LSTM layer as well as a dense layer with a dropout mechanism between LSTM layers to minimize the possibility of the model overfitting to the training data. A dropout mechanism randomly discards some neurons which introduces random error into the model, therefore preventing overfitting. Each neuron has a specified probability of being dropped in each layer. Below is a visual representation of the model.



```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 200, 32)           160000

 spatial_dropout1d (SpatialD  (None, 200, 32)          0
 ropout1D)

 lstm (LSTM)                 (None, 50)                16600

 dropout (Dropout)           (None, 50)                0

 dense (Dense)               (None, 1)                 51

=================================================================
Total params: 176,651
Trainable params: 176,651
Non-trainable params: 0
_________________________________________________________________
None
```

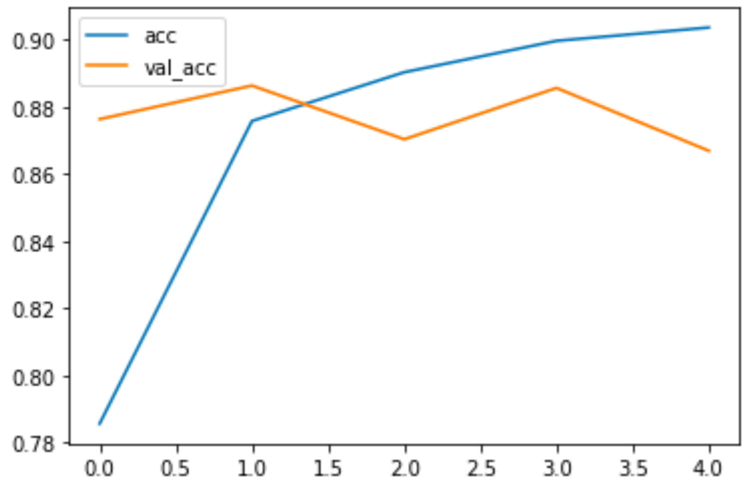
For the training of the model, we decided on using 5 epochs with a batch size of 32 samples where 20 percent of the data was retained as a validation data set. The training process took around 15 minutes. Visualized below are the training and validation accuracy metrics for the model.



```
Epoch 1/5
875/875 [==============================] - 169s 187ms/step - loss: 0.4493 - accuracy: 0.7856 - val_loss: 0.3184 - val_accuracy: 0.8763
Epoch 2/5
875/875 [==============================] - 158s 180ms/step - loss: 0.3033 - accuracy: 0.8757 - val_loss: 0.2784 - val_accuracy: 0.8863
Epoch 3/5
875/875 [==============================] - 155s 178ms/step - loss: 0.2750 - accuracy: 0.8903 - val_loss: 0.3411 - val_accuracy: 0.8703
Epoch 4/5
875/875 [==============================] - 155s 177ms/step - loss: 0.2543 - accuracy: 0.8996 - val_loss: 0.2826 - val_accuracy: 0.8856
Epoch 5/5
875/875 [==============================] - 156s 178ms/step - loss: 0.2403 - accuracy: 0.9036 - val_loss: 0.3814 - val_accuracy: 0.8669
```

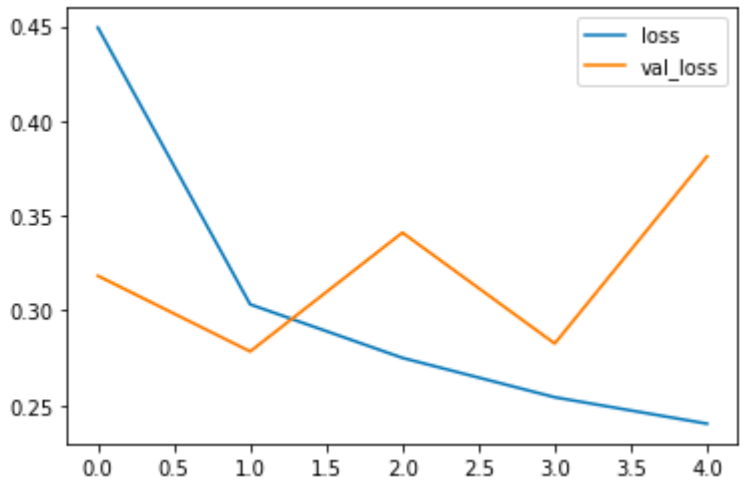
Overall, our model was able to accurately classify 90 percent of the training data and 86 percent of the validation data set which could potentially indicate a small amount of overfitting in the model's performance.

4. Analysis of the Model



This chart is a visual analysis of the accuracy metric. As we can see, accuracy of the model increases with every epoch. This makes sense because the network is able to learn the data in different ways with each passing epoch.

Juxtaposed to this, we have the loss metric visualised



As we can clearly see, the loss falls quickly and starts flattening out after 1.0 This happens as the accuracy of the model increases and the loss is minimised.

5. Further Applications

This project was interesting and useful as we can get a bird's eye view of what people are saying about a movie and understand if its a good movie or not. We can use the same algorithm for a variety of different tasks. For example, we can analyze tweets and categorize them as positive or negative. Google maps does something similar with their ratings. They analyze the reviews, and pick out certain words that are repeated in different reviews. They categorize these as positive or negative and incorporate that into the rating of a place. If we were tk extend this project, we could do something similar where we construct a tf-idf matrix and find words that accurately describe a movie.