D213: Advanced Data Analytics

Performance Assessment

Task 2

Logan Rosemeyer

Western Governor's University

D213: Advanced Data Analytics

Dr. Festus Elleh

April 12, 2023

Contents

A1: Research Question

My research question is: Can we predict whether someone feels favorably about a product, movie/tv show, or place of business based on the word choices in their review?

A2: Objectives and Goals

The goal of the data analysis is to predict whether someone feels favorably about a product, movie/tv show, or place of business based on the word choices in their review.

A3: Prescribed Network

One type of neural network that is capable of performing a text classification task for predictions would be a Recurrent Neural Network (RNN). RNNs are commonly used in natural language processing. (IBM, n.d.)

B1: Data Exploration

The first thing I had to do was to import my data from the three sources, as well as bind them all together. Then, I needed to get rid of any emojis or non-English characters. I did this by using the following code.

```
amazon = pd.read_csv("D:\MS DA\D213\Task 2\sentiment labelled
sentences\\amazon_cells_labelled.txt", sep = "\t", header=None,
names=["text", "score"]);\
amazon.shape


imdb = pd.read_csv("D:\MS DA\D213\Task 2\sentiment labelled
sentences\\imdb_labelled.csv", header=None, names=["text", "score"]);\
imdb.head


yelp = pd.read_csv("D:\MS DA\D213\Task 2\sentiment labelled
sentences\\yelp_labelled.csv", header=None, names=["text", "score"]);\
yelp.head


reviews = pd.concat([amazon, imdb, yelp])
reviews.dropna(inplace=True)
reviews.head

reviews['score'] = reviews['score'].astype(int)
reviews.head
reviews.dtypes


reviews['text'] = reviews['text'].str.replace("'", "")
reviews['text'] = reviews['text'].str.replace(r'[^\w\s]+', ' ')
reviews['text'] = reviews['text'].str.replace('\d+', '')
reviews['text'] = reviews['text'].str.replace('  ', ' ')
reviews['text'] = reviews['text'].str.lower()
display(reviews.head)
```

My vocabulary size is 4996, which I got from the following code.

```
all_words = tfidf.fit_transform(reviews['final'])
print(all_words.shape)
```

Based on this, my proposed word embedding length is 9, which is the fourth root of my vocabulary size.

My maximum sequence length is 45, which I found using the following code.

```
max_words = max([len(t_line) for t_line in reviews['nostop_text']])
max_words
```

This length will allow us to preserve the available input data so that the generated model doesn't yield conclusions that are unlikely to generalize well.

B2: Tokenization

The goal of the tokenization process is to split the reviews into smaller units that be assigned meaning more easily (Burchfiel, 2022). I used the following code to tokenize my data.

```
def tokenize(text):
    words = re.split('\W+', text)
    return words

reviews['token_text'] = reviews['text'].apply(lambda x: tokenize(x))
reviews.head()
```

B3: Padding Process

Padding was not necessary for this process. I was able to load my pandas dataframe to tensorflow. This means that I do not have to worry about the strings having the same length. I used the following code to do this.

```
tensor_dataset = tf.data.Dataset.from_tensor_slices((features.values,
y.values))

train_tf_dataset = tensor_dataset.shuffle(len(reviews)).batch(1)
```

B4: Categories of Sentiment

I have two categories of sentiment, positive and negative. A two represents positive sentiment, while a two represents a negative sentiment.

B5: Steps to Prepare the Data

The first thing I did to prepare the data was to read the files in and concatenate all rows using the following code.

```
amazon = pd.read_csv("D:\MS DA\D213\Task 2\sentiment labelled
sentences\\amazon_cells_labelled.txt", sep = "\t", header=None,
names=["text", "score"]);\
amazon.shape


imdb = pd.read_csv("D:\MS DA\D213\Task 2\sentiment labelled
sentences\\imdb_labelled.csv", header=None, names=["text", "score"]);\
imdb.head


yelp = pd.read_csv("D:\MS DA\D213\Task 2\sentiment labelled
sentences\\yelp_labelled.csv", header=None, names=["text", "score"]);\
yelp.head


reviews = pd.concat([amazon, imdb, yelp])
reviews.dropna(inplace=True)
reviews.head
```

Then, I changed the score column to an integer type.

```
reviews['score'] = reviews['score'].astype(int)
reviews.head
reviews.dtypes
```

After that, I got rid of any special characters and made all reviews lowercase.

```
reviews['text'] = reviews['text'].str.replace("'", "")
reviews['text'] = reviews['text'].str.replace(r'[^\w\s]+', ' ')
reviews['text'] = reviews['text'].str.replace('\d+', '')
reviews['text'] = reviews['text'].str.replace('  ', ' ')
reviews['text'] = reviews['text'].str.lower()
display(reviews.head)
```

Finally, I split the data into training, validation, and test split. Out of the 2995 reviews, 2545 were included in my training set, 599 in my test set, and 450 in my validation set. I used the following code to split the data.

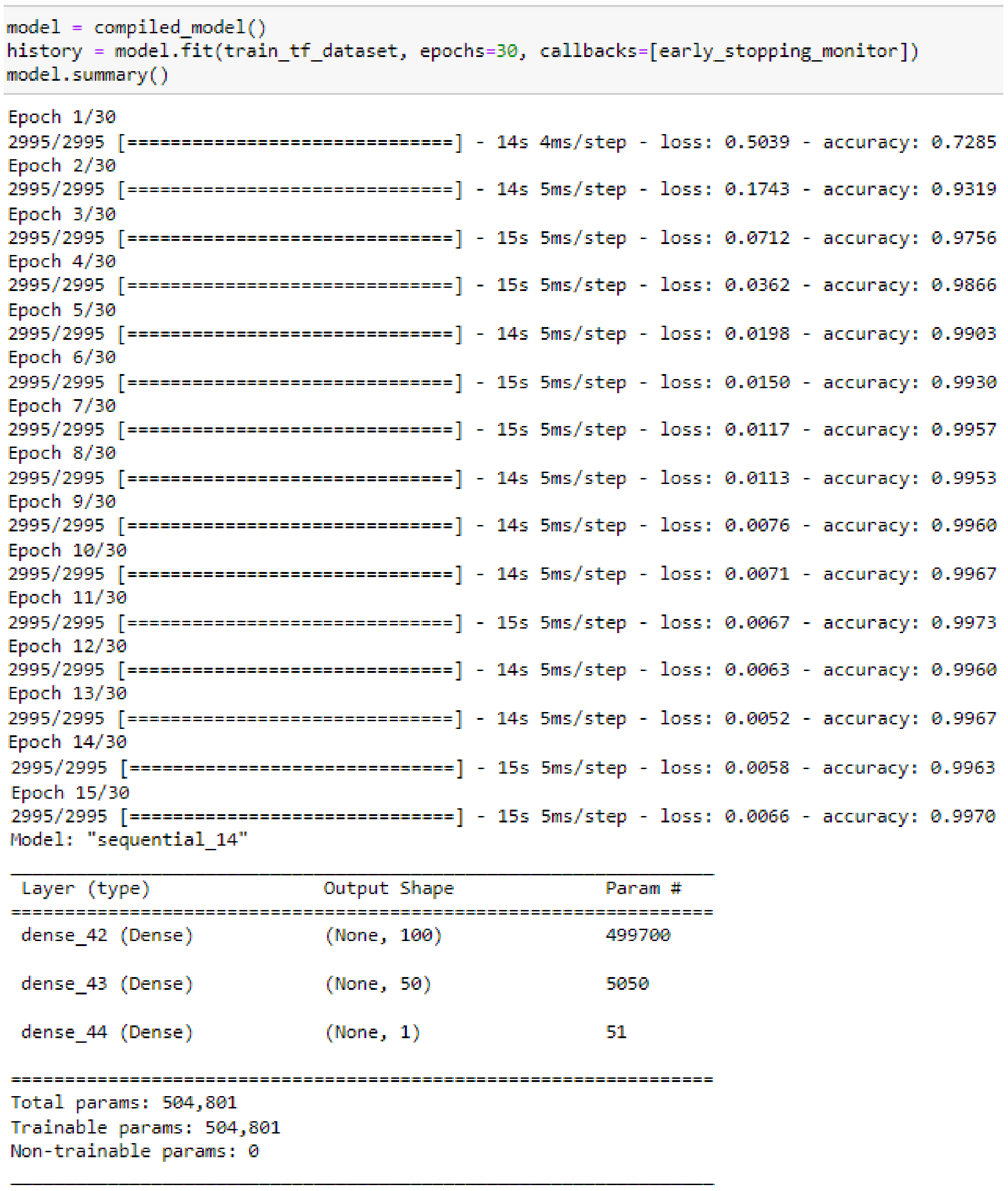
```
x = features

y = reviews['score']


x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2,
random_state=22)

x_train, x_val, y_train, y_val = train_test_split(x, y, test_size=0.15,
random_state=22)


print("Training size: ", x_train.shape)

print("Test size: ", x_test.shape)
print("Validation size: ", x_val.shape)
```

B6: Prepared Dataset

The prepared data is included in my submission.

C1: Model Summary

Below is a screenshot of my model summary.



```
model = compiled_model()
history = model.fit(train_tf_dataset, epochs=30, callbacks=[early_stopping_monitor])
model.summary()
```

```
Epoch 1/30
2995/2995 [==============================] - 14s 4ms/step - loss: 0.5039 - accuracy: 0.7285
Epoch 2/30
2995/2995 [==============================] - 14s 5ms/step - loss: 0.1743 - accuracy: 0.9319
Epoch 3/30
2995/2995 [==============================] - 15s 5ms/step - loss: 0.0712 - accuracy: 0.9756
Epoch 4/30
2995/2995 [==============================] - 15s 5ms/step - loss: 0.0362 - accuracy: 0.9866
Epoch 5/30
2995/2995 [==============================] - 14s 5ms/step - loss: 0.0198 - accuracy: 0.9903
Epoch 6/30
2995/2995 [==============================] - 15s 5ms/step - loss: 0.0150 - accuracy: 0.9930
Epoch 7/30
2995/2995 [==============================] - 15s 5ms/step - loss: 0.0117 - accuracy: 0.9957
Epoch 8/30
2995/2995 [==============================] - 14s 5ms/step - loss: 0.0113 - accuracy: 0.9953
Epoch 9/30
2995/2995 [==============================] - 14s 5ms/step - loss: 0.0076 - accuracy: 0.9960
Epoch 10/30
2995/2995 [==============================] - 14s 5ms/step - loss: 0.0071 - accuracy: 0.9967
Epoch 11/30
2995/2995 [==============================] - 15s 5ms/step - loss: 0.0067 - accuracy: 0.9973
Epoch 12/30
2995/2995 [==============================] - 14s 5ms/step - loss: 0.0063 - accuracy: 0.9960
Epoch 13/30
2995/2995 [==============================] - 14s 5ms/step - loss: 0.0052 - accuracy: 0.9967
Epoch 14/30
2995/2995 [==============================] - 15s 5ms/step - loss: 0.0058 - accuracy: 0.9963
Epoch 15/30
2995/2995 [==============================] - 15s 5ms/step - loss: 0.0066 - accuracy: 0.9970
Model: "sequential_14"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_42 (Dense)            (None, 100)               499700

 dense_43 (Dense)            (None, 50)                5050

 dense_44 (Dense)            (None, 1)                 51

=================================================================
Total params: 504,801
Trainable params: 504,801
Non-trainable params: 0
_________________________________________________________________
```

C2: Network Architecture

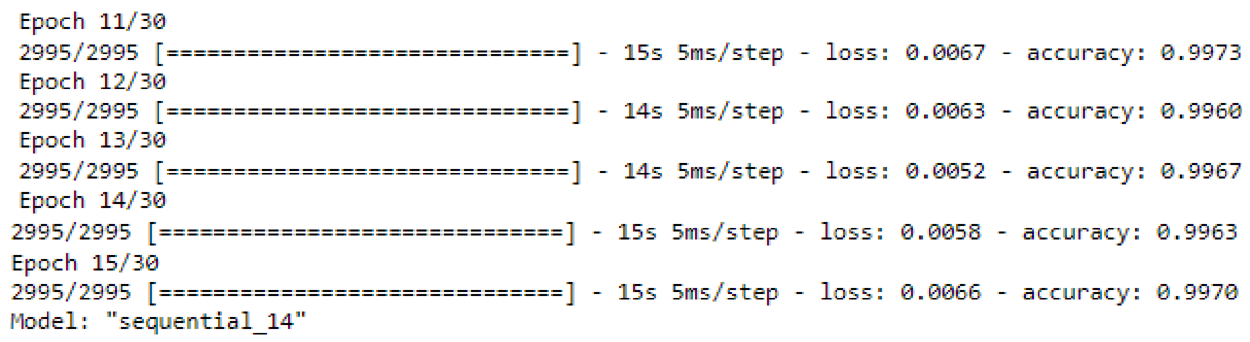
There are three layers in my model. They are all dense layers. The first layer has 499700 parameters, the second layer has 5050 parameters, the third has 51 parameters. This gives us a total of 504801 parameters.

C3: Hyperparameters

I used relu as my activation function. I chose relu because it has the ability to learn complex patterns in the data, and text would be considered a complex pattern. After trying a few different numbers for my nodes in each layer, I found that 100 nodes in the first layer, 50 in the second layer, and 2 in the last layer gave me very accurate results, over 99%. I used binary cross-entropy. This is perfect for my model because it needs to classify the results into one of two categories, either positive or negative reviews. I chose to use the adam optimizer because it seems to be the most popular adaptive optimizer in most cases. There is also no need to focus on the learning rate (Giordano, 2020). I included an early stopping monitor to prevent overfitting where we monitor loss and set the patience to 2. I used accuracy as the metric to evaluate my model.

D1: Stopping Criteria

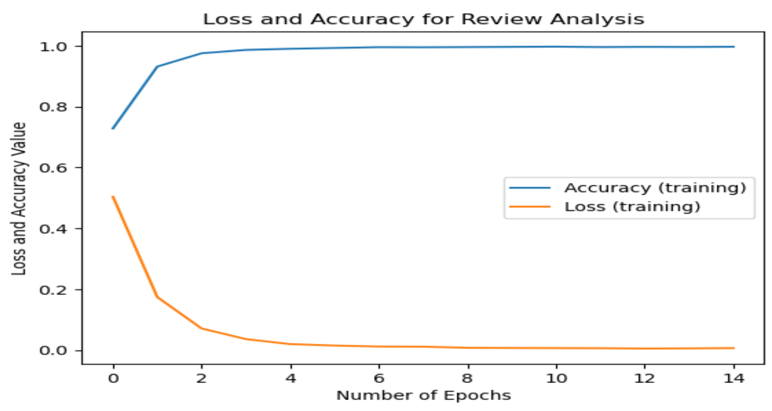
Using stopping criteria has a huge impact on the model, because it prevents over training. This allows us to run our model without having to guess and check which amount of epochs will give us the best model. Below is a screenshot of the last few training epochs.



```
Epoch 11/30
2995/2995 [==============================] - 15s 5ms/step - loss: 0.0067 - accuracy: 0.9973
Epoch 12/30
2995/2995 [==============================] - 14s 5ms/step - loss: 0.0063 - accuracy: 0.9960
Epoch 13/30
2995/2995 [==============================] - 14s 5ms/step - loss: 0.0052 - accuracy: 0.9967
Epoch 14/30
2995/2995 [==============================] - 15s 5ms/step - loss: 0.0058 - accuracy: 0.9963
Epoch 15/30
2995/2995 [==============================] - 15s 5ms/step - loss: 0.0066 - accuracy: 0.9970
Model: "sequential_14"
```

D2: Training Process

The following visualization shows the accuracy and loss as the number of epochs increases.



D3: Fit

The fitness of my model is excellent. The accuracy on my test and validation datasets is over 99%. One reason that I was able to avoid overfitting the data was that I incorporated stopping criteria into my model.

D4: Predictive Accuracy

My model is extremely accurate. Both the validation and test datasets were predicted with an accuracy over 0.99 and loss of less than 0.005.

E: Code

A copy of the code has been provided in the submission.

F: Functionality

My model input 2545 reviews to train, then validated with 450 reviews, and tested with 599 reviews. I used NLP to analyze the sentiments of the customers and predicted whether their review was positive or negative. The model ended up being extremely functional, with an accuracy on all datasets of over 99%.

G: Recommendations

I would recommend that the organization use my model to predict whether a review is positive or negative. This will help us to get a better idea of whether customers like or dislike our product, as well as pinpoint which words tend to lead to positive reviews, and which words will lead to negative reviews.

H: Reporting

An HTML document of my executed notebook is included in the submission.

I: Sources for Third-Party Code

Jedamski, Derek. *Advanced NLP with Python for Machine Learning*. LinkedIn Learning. Retrieved April 18, 2023.
https://www.linkedin.com/learning/advanced-nlp-with-python-for-machine-learning/cleaning-text-data?autoplay=true&resume=false&u=2045532

Ydobon, A. *[Tensorflow 2.0] Load Pandas dataframe to Tensorflow.* Medium. September 26, 2019. Retrieved April 18, 2023.
https://financial-engineering.medium.com/tensorflow-2-0-load-pandas-dataframe-to-tensorflow-202c5d48966b

J: Sources

Burchfiel, Anni. *What is NLP (Natural Language Processing) Tokenization?.* Tokenex. May 16, 2022. Retrieved April 16, 2023.

https://www.tokenex.com/blog/ab-what-is-nlp-natural-language-processing-tokenization/#:~:text=Tokenization%20is%20used%20in%20natural,into%20understandable%20parts%20(words).

What are recurrent neural networks?. IBM. Retrieved April 18, 2023.
https://www.ibm.com/topics/recurrent-neural-networks

Giordano, Davide. *7 tips to choose the best optimizer.* Towards Data Science. July, 25, 2020. Retrieved April 20, 2023.
https://towardsdatascience.com/7-tips-to-choose-the-best-optimizer-47bb9c1219e