Raquel Ocasio

D213 – Advanced Data Analytics, Task 2: Sentiment Analysis Using Neural Networks

April 23, 2024

Western Governors University

Part I: Research Questions

A1

Using previous movie reviews from other users, can we predict a movie review as either positive or negative?

A2

The goal of the data analysis is to attempt to predict if a movie review will be positive or negative based on the word choices of the reviewer.

A3

The Recurrent Neural Network (RNN) is a type of neural network that can be trained to identify frequently used sentiments in a dataset of movie reviews.

Part II: Data Preparation

B1

The data preparation process included performing exploratory data analysis (EDA), checking for and removing unusual characters, converting text to lowercase, calculating vocabulary size, determining word embedding length, and choosing a maximum sequence length. EDA revealed that the dataset had duplicate values which were removed.

Unusual characters include items such as punctuation marks, emojis, and non-English characters. These were detected using a regular expression to match the dataset against common English alphanumeric characters.

Vocabulary size is the number of unique words in the dataset.

A word embedding represents the position of a word within a learned vector space. The word embedding length is determined by taking the fourth root of the vocabulary size and rounding up to the nearest integer.

The maximum sequence length helps to maintain the integrity of the input data, ensuring that the resulting model avoids drawing conclusions that may not generalize effectively. It is determined by taking the average of the movie review length and rounding up to the nearest integer. Inputs shorter than the maximum sequence length can be handled through padding, while inputs longer than the maximum sequence length can be handled with truncation.

B2

The goal of the tokenization process is to divide text into smaller pieces so that each piece can be assigned an index. The sub() and compile() methods from the Regular Expressions library (re) were used to normalize the text.

```
*** View dataset with tokens column ***
                                            sentence  score  \
0  A very, very, very slow-moving, aimless movie ...      0
1  Not sure who was more lost - the flat characte...      0
2  Attempting artiness with black & white and cle...      0
3       Very little music or anything to speak of.        0
4  The best scene in the movie was when Gerardo i...      1

                                      tokenized_text
0  [3, 28, 28, 28, 287, 407, 1216, 12, 37, 3, 121...
1  [25, 522, 52, 11, 61, 409, 1, 735, 57, 46, 1, ...
2  [1220, 1221, 20, 210, 233, 2, 337, 185, 738, 1...
3                 [28, 117, 129, 46, 234, 9, 523, 4]
4  [1, 73, 130, 10, 1, 12, 11, 64, 1223, 5, 410, ...
```

B3

The length of the sequences was standardized by padding using the sequence.pad_sequences() function. The function used the value for the chosen maximum sequence length to calculate the length of the sequence. The padding occurs at the end of each sequence. A screenshot of the a single padded sequence is below.

```
*** View single padded sequence ***
[65, 24, 24, 24, 284, 403, 1196, 10, 33, 1197, 1198, 404, 139, 0, 0, 0, 0, 0, 0, 0]
```

B4

Two categories of sentiment will be used: negative and positive. An activation function for the final dense layer of the network is softmax. The softmax function is used to calculate the relative probabilities.

B5

The following steps were taken to prepare the data for analysis:

1. Read the CSV file into a dataframe.
2. Checked the structure of the dataset.
3. Cleaned the dataset by removing any duplicates, removing any rows with missing values, removing unusual characters (HTML, punctuation, numbers, extra spaces, etc.), and converting all text to lowercase.
4. Tokenized the text.
5. Calculated the maximum sequence length for padding.
6. Padded the tokenized text.
7. Split the data into train, test, and validation sets using a 70/15/15 ratio. (Acharya, 2023). The size of the train, test, and validation sets are 521, 112, and 112, respectively.

B6

See attached file.

Part III: Network Architecture

C1

Layer (type)	Output Shape	Param #
embedding_1 (Embedding)	(None, 17, 32)	70,400
global_average_pooling1d_1 (GlobalAveragePooling1D)	(None, 32)	0
dense_3 (Dense)	(None, 64)	2,112
dropout_1 (Dropout)	(None, 64)	0
dense_4 (Dense)	(None, 32)	2,080
dense_5 (Dense)	(None, 1)	33

Total params: 223,877 (874.52 KB)

Trainable params: 74,625 (291.50 KB)

Non-trainable params: 0 (0.00 B)

Optimizer params: 149,252 (583.02 KB)

C2

The model has six layers. The first layer is Embedding, of type input. The first layer has 70,400 parameters.

The second layer is GlobalAveragePooling1D, of type pooling. The second layer has zero parameters.

The third layer is Dense, of type hidden. The third layer has 2,112 parameters.

The fourth layer is Dropout, of type dropout. The fourth layer has zero parameters.

The fifth layer is Dense, of type hidden. The fifth layer has 2,080 parameters.

The sixth layer is Dense, of type output. The sixth layer has 33 parameters.

The total number of parameters in the network is 223,877.

C3-a

The relu function was used in the third and fifth Dense hidden layers to overcome the limitations of other activation functions.

The Sigmoid function was used as an activation function in the sixth Dense output layer for binary classification.

C3-b

The first layer (Embedding) has 70,400 nodes.

The second layer (GlobalAveragePooling1D) doesn't change the number of nodes. It performs pooling across the time dimension and outputs a fixed-size tensor.

The third layer (first Dense) has 2,048 nodes.

The fourth layer (Dropout) doesn't change the number of nodes. It applies dropout regularization, randomly dropping out a fraction of input units during training.

The fifth layer (second Dense) has 2,048 nodes.

The sixth layer (output Dense) has 32 nodes.

C3-c

The model uses binary crossentropy as a loss function for categorizing data that operates on binary classification.

C3-d

The model uses the Adaptive Moment Estimation (adam) algorithm as an optimizer function. The adam algorithm helps with calculating adaptive individual learning rates for different parameters during the model training process. (Agarwal, 2023)

C3-e

The model uses early stopping criteria so that overfitting of the training data doesn't occur. It also stops running the model on the training data when the validation score stops improving after three epochs.

C3-f

The model uses accuracy as an evaluation metric. The accuracy metric determines if the model is at risk for overfitting.

Part IV: Model Evaluation

D1

The number of epochs impacts the use of stopping criteria by potentially providing enough opportunities for the model to find a stopping point during the training phase at which the validation score no longer improves.

```
Epoch 18/35
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.9970 - loss: 0.0640 - val_accuracy: 0.8304 - val_loss: 0.5240
```

D2

The improvement in training accuracy and decrease in training loss indicate that the model is learning. Concurrently, the consistent validation accuracy and loss throughout training epochs suggest that the model is generalizing well. As the model consistently performs well on both training and validation data across different runs, it suggests robustness.

To address overfitting, the model uses early stopping criteria while gradually decreasing, and then increasing the number of epochs to 35 for optimal accuracy. Screenshots of the accuracy results at 40, 30, and 35 epochs are below.

Results at 40 epochs

```
Epoch 17/40
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 6ms/step - accuracy: 0.9914 - loss: 0.0892 - val_accuracy: 0.7857 - val_loss: 0.5436
```

Results at 30 epochs

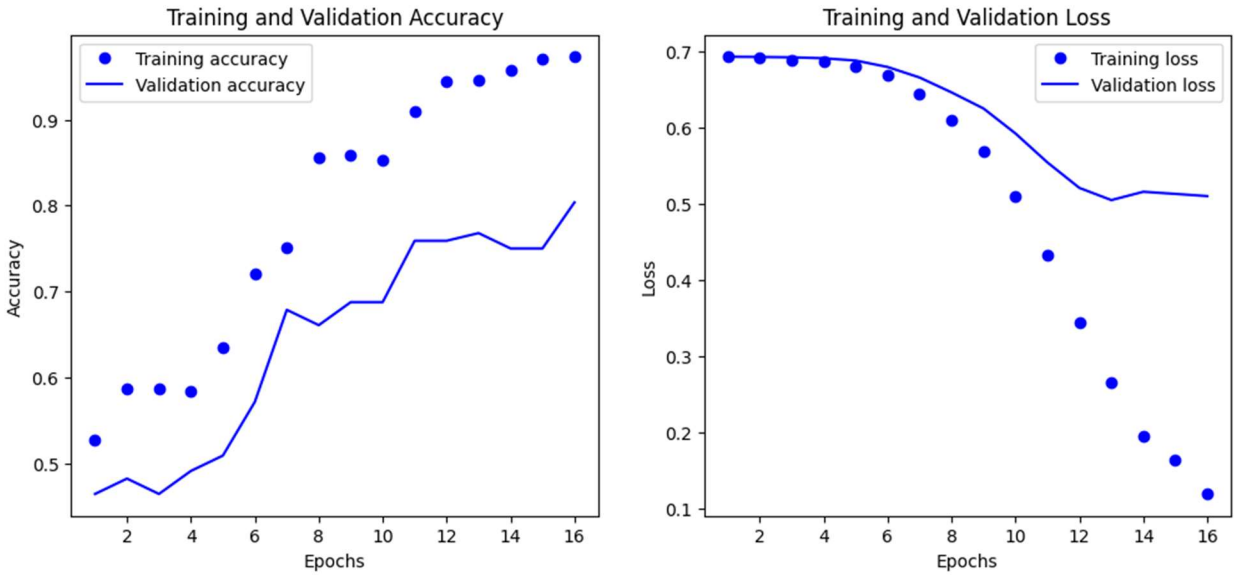
```
Epoch 16/30
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step - accuracy: 0.9848 - loss: 0.1059 - val_accuracy: 0.7768 - val_loss: 0.5356
```

Results at 35 epochs (final model)

```
Epoch 18/35
9/9 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.9970 - loss: 0.0640 - val_accuracy: 0.8304 - val_loss: 0.5240
```

The performance of the final model on the test data closely mirrored its performance on the training data. Achieving an accuracy score of 99.7% on the training data and 83.0% on the test data suggests minimal overfitting, as the model's performance remained consistent across both datasets.

D3



D4

The model achieved 99.7% accuracy on the training data with a loss of 0.064, 83% accuracy on the testing data with a loss of 0.524, and 73.6% accuracy on the validation data with a loss of 0.564. While accuracy decreased and loss increased on the test and validation data, the model appears to be a good fit overall.

Part V: Summary and Recommendations

E

```
# Part E
# Define ModelCheckpoint callback to save the best model
model_checkpoint = tf.keras.callbacks.ModelCheckpoint
(filepath="best_model.keras", save_best_only=True)
```

F

The functionality of the neural network is to determine if a movie review can be classified as positive or negative. The model was trained with a dataset of 521 movie reviews and their actual sentiments recorded as labels. The model was tested and validated with dataset sizes of 112 each. The network architecture has had the impact of making it possible to determine whether a review is negative or positive with a high degree of accuracy.

G

The recommendation to improve the accuracy of the model is to use larger datasets for training, testing, and validation.

Part VI: Reporting

H

See attached py and HTML files.

I

No web sources were used to acquire segments of code.

J

Dataset:

Kotzias,Dimitrios. (2015). Sentiment Labelled Sentences. UCI Machine Learning Repository. https://doi.org/10.24432/C57604.

Sources:

Acharya, A. (2023, June 13). Training, validation, test split for Machine Learning Datasets. Encord. https://encord.com/blog/train-val-test-split/

Agarwal, R. (2023, September 23). Complete guide to the adam optimization algorithm. Built In. https://builtin.com/machine-learning/adam-optimization

(Agarwal, 2023)