D213 Sentiment Analysis Using Neural Networks

Darien Nguyen

Western Governors University

D213 Task Two: Sentiment Analysis Using Neural Networks

July 8, 2023

Part I: Research Question

A. Describe the purpose of this data analysis by doing the following:

1. Summarize one research question that you will answer using neural network models and NLP techniques. Be sure the research question is relevant to a real-world organizational situation and sentiment analysis captured in your chosen data set(s).

Can we accurately predict the sentiment (positive or negative) of movie reviews using a neural network model on a given dataset of movie reviews?

2. Define the objectives or goals of the data analysis. Be sure the objectives or goals are reasonable within the scope of the research question and are represented in the available data.

The objectives of the data analysis are as follows:

* Can we develop and train a neural network model capable of sentiment analysis on movie reviews?
* How accurate is the model in predicting sentiment labels (positive or negative) based on the given dataset?

3. Identify a type of neural network capable of performing a text classification task that can be trained to produce useful predictions on text sequences on the selected data set.

To perform text classification tasks like sentiment analysis on the given dataset of movie reviews, we will utilize a type of neural network known as a recurrent neural network (RNN). Specifically, we will use a variant of RNN called Long Short-Term Memory (LSTM) network. TensorFlow, as a powerful deep learning framework, provides the necessary tools and functionalities to implement LSTM-based models for text classification tasks.

Part II: Data Preparation

B. Summarize the data cleaning process by doing the following:

To summarize the data cleaning process for the movie reviews dataset, the following steps were taken:

1. Data Loading: The dataset, named "imdb\_labelled.txt," was loaded from the file. Each line in the file contains a movie review and its corresponding sentiment label (0 for negative and 1 for positive).
2. Text Preprocessing: The text data in the movie reviews was preprocessed to prepare it for model training. The following preprocessing steps were performed:
   * Convert to Lowercase: All text was converted to lowercase to ensure uniformity in text representations.
   * Remove Special Characters and Digits: Special characters and digits were removed from the text using regular expressions. This step helped to clean the text and remove irrelevant information that may not contribute to sentiment analysis.
3. Tokenization: The text data was tokenized using the TensorFlow Tokenizer. Tokenization involves converting each text review into a sequence of integer tokens, where each token represents a unique word in the review. This step is essential for feeding the text data into the neural network.
4. Padding Sequences: To create a fixed-length input for the neural network, the tokenized sequences were padded or truncated to ensure they all have the same length. The maximum sequence length of the reviews was determined, and sequences shorter than this length were padded with zeros.
5. Data Split (Not Mentioned in the provided code snippet): It's important to mention that the data should be split into training and testing sets before training the model. Typically, the dataset is divided into a training set (e.g., 80% of the data) and a testing set (e.g., 20% of the data). The training set is used to train the model, while the testing set is used to evaluate its performance on unseen data.

1. Perform exploratory data analysis on the chosen data set, and include an explanation of each of the following elements:

• presence of unusual characters (e.g., emojis, non-English characters)

Elements of unusual characters detected and removed. Characters were found and relayed back out as ascii values seen below:

Unusual characters found:

{'\x96', 'å', '\x85', 'é', '\x97'}

• vocabulary size

To track the amount of unique words present found within the data set.

Vocabulary Size: 2785

• proposed word embedding length

Best embedding length: 200

• statistical justification for the chosen maximum sequence length

Training model with embedding length: 200

Epoch 1/50

10/10 [==============================] - 5s 287ms/step - loss: 1.1586 - accuracy: 0.4969 - val\_loss: 1.1162 - val\_accuracy: 0.5188

Epoch 2/50

10/10 [==============================] - 2s 251ms/step - loss: 1.0752 - accuracy: 0.6812 - val\_loss: 1.0394 - val\_accuracy: 0.5813

Epoch 3/50

10/10 [==============================] - 2s 248ms/step - loss: 0.9766 - accuracy: 0.7391 - val\_loss: 0.9532 - val\_accuracy: 0.6250

Epoch 4/50

10/10 [==============================] - 2s 242ms/step - loss: 0.8087 - accuracy: 0.8297 - val\_loss: 0.8236 - val\_accuracy: 0.7563

Epoch 5/50

10/10 [==============================] - 3s 289ms/step - loss: 0.5684 - accuracy: 0.9031 - val\_loss: 0.8165 - val\_accuracy: 0.7312

Epoch 6/50

10/10 [==============================] - 3s 291ms/step - loss: 0.4259 - accuracy: 0.9484 - val\_loss: 0.7768 - val\_accuracy: 0.7688

Epoch 7/50

10/10 [==============================] - 3s 289ms/step - loss: 0.3153 - accuracy: 0.9781 - val\_loss: 0.6991 - val\_accuracy: 0.8062

Epoch 8/50

10/10 [==============================] - 3s 289ms/step - loss: 0.2593 - accuracy: 0.9859 - val\_loss: 0.7386 - val\_accuracy: 0.8062

Epoch 9/50

10/10 [==============================] - 3s 288ms/step - loss: 0.2181 - accuracy: 0.9922 - val\_loss: 0.7056 - val\_accuracy: 0.7937

Epoch 10/50

10/10 [==============================] - 3s 285ms/step - loss: 0.1856 - accuracy: 0.9969 - val\_loss: 0.7507 - val\_accuracy: 0.7812

7/7 [==============================] - 0s 15ms/step - loss: 0.7944 - accuracy: 0.7700

Validation Accuracy with embedding length 200: 0.7699999809265137

Embedding model 200 showcased the highest accuracy in relation to embedding length in comparison to the others tested in the code.

2. Describe the goals of the tokenization process, including any code generated and packages that are used to normalize text during the tokenization process.

The goals of our tokenization process in the NLP were:

1. **Convert Text to Numerical Representation**: Tokenization converts the raw text data into a numerical representation, where each word is represented by a unique integer index. This is necessary because most machine learning models, including neural networks, require numerical inputs to perform computations.
2. **Build Vocabulary**: The tokenization process builds a vocabulary of words from the training data. The vocabulary contains a mapping of each unique word in the corpus to a corresponding index. This vocabulary is essential for mapping words to integers during tokenization and for later word embedding.
3. **Normalization**: The tokenization process includes text normalization, which involves converting all text to lowercase and removing any special characters and digits. This step helps in reducing the vocabulary size, removing noise, and ensuring that similar words with different cases or punctuation are treated as the same token.
4. **Prepare Data for Deep Learning**: Tokenization prepares the text data to be used as input for training deep learning models, such as the recurrent neural network (RNN) used in the provided code. By tokenizing the text, we create sequences of integer tokens that can be fed into the neural network's embedding layer.
5. **Enable Sequence Padding**: Tokenization allows us to determine the maximum sequence length in the dataset, which is essential for padding the sequences to ensure that all inputs to the neural network have the same length. This is necessary because neural networks typically require fixed-size inputs.

3. Explain the padding process used to standardize the length of sequences. Include the following in your explanation:

• if the padding occurs before or after the text sequence

The padding occurs after the text sequence. **tf.keras.preprocessing.sequence.pad\_sequences()** pads the sequences with zeros (0s) at the end of each sequence until they all have the same length. This is the default behavior of the function. Padding at the end ensures that the original content of the text is preserved, and the zeros are added to the end of the sequence to reach the desired fixed length.

• a screenshot of a single padded sequenceA screenshot of a computer

Description automatically generated

4. Identify how many categories of sentiment will be used and an activation function for the final dense layer of the network.

We will be using two categories of sentiment, 1 and 0 for positive and negative. For the final dense layer of the network, the activation function used is the sigmoid function. The sigmoid activation function is commonly used in binary classification problems because it squashes the output values between 0 and 1, representing the probability of the input belonging to the positive class. It is well-suited for this type of task as it can produce a probability score that can be directly interpreted as the likelihood of the input belonging to the positive class.

5. Explain the steps used to prepare the data for analysis, including the size of the training, validation, and test set split (based on the industry average).

1. **Loading and Cleaning Data**: The data is loaded from the 'imdb\_labelled.txt' file, containing text reviews and their corresponding sentiment labels. Text is converted to lowercase, and a cleaning function is applied to remove special characters and digits.
2. **Tokenization**: The text data is tokenized using the Keras Tokenizer, which breaks down the text into individual words (tokens) and assigns unique integer IDs to each token. The tokenizer is fitted on the training data to create a vocabulary of unique words and their corresponding integer IDs.
3. **Padding Sequences**: The sequences of tokens are padded to a fixed length using the **pad\_sequences** function from Keras. Padding ensures that all sequences have the same length, which is determined as the maximum sequence length among all sequences in the training data. Padding is done at the end of the sequence.
4. **Data Split**: The data is split into training and validation sets using the **train\_test\_split** function from the **sklearn.model\_selection** module. The data is split in an 80-20 ratio, where 80% of the data is used for training and 20% for validation.
5. **Model Architecture**: The code defines a function **create\_model** to construct the neural network model for sentiment analysis. The model includes an Embedding layer, a Bidirectional LSTM layer, a Dense layer with Rectified Linear Unit (ReLU) activation, and a final Dense layer with a sigmoid activation function for binary classification.
6. **Training and Evaluation**: The code defines a function **train\_and\_evaluate\_model** to train and evaluate the model with different word embedding lengths. The model is trained on the training data using binary cross-entropy loss and the Adam optimizer. It is then evaluated on the validation data to measure its performance.
7. **Choosing Best Embedding Length**: The code loops through different embedding lengths and selects the best embedding length based on validation accuracy. The goal is to determine the word embedding length that yields the highest accuracy on the validation set.
8. **Histogram of Sequence Lengths**: The code calculates the sequence lengths for all padded sequences and plots a histogram to visualize the distribution of sequence lengths. This helps in understanding the distribution of text lengths in the dataset.
9. **Printing Padded Sequence**: The code prints the original sequence and the corresponding padded sequence for the first review in the training data. This demonstrates how padding is applied to standardize the sequence lengths.

6. Provide a copy of the prepared data set.

Part III: Network Architecture

C. Describe the type of network used by doing the following:

1. Provide the output of the model summary of the function from TensorFlow.

A screenshot of a computer

Description automatically generated

2. Discuss the number of layers, the type of layers, and the total number of parameters.

1. Embedding Layer: The first layer is an Embedding layer, which is used to convert the input text data into dense word vectors. In this model, the embedding layer has 826,200 trainable parameters, indicating that the model is learning to represent each word in a 300-dimensional vector space.
2. Bidirectional LSTM Layer: The second layer is a Bidirectional LSTM layer, which is a type of recurrent neural network (RNN) that processes the input sequence in both forward and backward directions. This allows the model to capture contextual information from both past and future tokens in the text. The bidirectional LSTM layer has 186,880 trainable parameters, and it is designed with 128 units.
3. Dense Layer: The third layer is a Dense layer with 4,128 trainable parameters. This layer is added to introduce non-linearity and reduce the dimensionality of the data. It uses the Rectified Linear Unit (ReLU) activation function, which helps the model learn complex patterns in the data.
4. Output Dense Layer: The final layer is an output Dense layer with 33 trainable parameters and a sigmoid activation function. This layer produces a binary sentiment prediction, indicating whether the movie review is positive or negative.
5. Total Number of Parameters: The model has a total of 1,017,241 parameters, all of which are trainable. These parameters are learned during the training process and represent the weights and biases of the different layers in the model. The total number of parameters indicates the model's capacity to learn from the data and make predictions. It is crucial to strike a balance between having enough parameters to capture the underlying patterns in the data and avoiding overfitting, where the model memorizes the training data but fails to generalize well to new, unseen data. Regularization techniques and early stopping are often used to prevent overfitting in deep learning models.

3. Justify the choice of hyperparameters, including the following elements:

• activation functions

* For the hidden layers, the ReLU (Rectified Linear Unit) activation function is commonly used. It introduces non-linearity and helps the model learn complex patterns in the data.
* The sigmoid activation function is used in the final dense layer. Since sentiment analysis is a binary classification task (positive or negative sentiment), the sigmoid function maps the output to a probability value between 0 and 1, making it suitable for binary classification problems.

• number of nodes per layer

* The number of nodes in each layer is determined through experimentation and hyperparameter tuning. In this case, 64 nodes were chosen for the dense layers based on empirical performance and to strike a balance between model complexity and performance.

• loss function

* For binary classification tasks like sentiment analysis, the binary cross-entropy loss function (binary\_crossentropy) is commonly used. It measures the difference between the predicted probability and the true label for each sample, providing a suitable loss metric for binary classification problems.

• optimizer

* The Adam optimizer is widely used in deep learning tasks due to its adaptive learning rate and momentum features. It helps speed up convergence and finds an appropriate learning rate for each parameter, making it suitable for training the model efficiently.

• stopping criteria

* Early stopping is often used as a stopping criterion during model training. It allows the training to stop early if there is no improvement in the validation performance for a certain number of epochs, preventing overfitting and saving computation time.

• evaluation metric

* The evaluation metric used for this model is accuracy. Accuracy is a common metric for classification tasks, and it measures the proportion of correct predictions to the total number of predictions. In sentiment analysis, accuracy provides a clear understanding of how well the model is performing in predicting sentiment polarity (positive or negative) on the test set.

Part IV: Model Evaluation

D. Evaluate the model training process and its relevant outcomes by doing the following:

1. Discuss the impact of using stopping criteria to include defining the number of epochs, including a screenshot showing the final training epoch.

We can see that the training process employed early stopping with a patience of 3 epochs. Early stopping monitors the validation loss and stops the training process if the validation loss does not improve for a certain number of epochs (patience). The purpose of using early stopping is to prevent overfitting and to find the best model parameters where the validation loss is minimal.

For embedding length 50, the training process stopped at epoch 11. Similarly, for embedding lengths 100 and 200, the training process stopped at epoch 10. For embedding length 300, the training process stopped at epoch 9. These early stopping points indicate that the model's performance on the validation set did not improve after the specified number of epochs.

Defining the number of epochs is a critical hyperparameter to avoid underfitting and overfitting. Underfitting occurs when the model hasn't learned enough from the data, and overfitting occurs when the model memorizes the training data too well and performs poorly on new data.

In the output, we can observe that the models were trained for 50 epochs. However, due to early stopping, the actual number of epochs used for training varies depending on the embedding length and when the early stopping criterion was met. For example, the model with an embedding length of 300 was trained for only 9 epochs.

The use of early stopping helped prevent overfitting in this scenario. For most embedding lengths, the training process stopped much earlier than the specified 50 epochs, indicating that the models found a good stopping point where the validation performance was best.

The final training epoch refers to the last epoch the model completes before the training process stops. The output shows the number of epochs the model was trained before early stopping was triggered.

In summary, early stopping is an effective technique to prevent overfitting and improve the generalization of the model. By stopping the training process early when the model's performance on the validation set plateaus, we can achieve better results with fewer training epochs. Defining the number of epochs is essential to control model training and avoid potential issues of underfitting or overfitting. In this case, early stopping allowed the models to reach optimal performance with fewer training epochs, saving computational resources and time.

2. Assess the fitness of the model and any actions taken to address overfitting.

We trained four different models with varying embedding lengths: 50, 100, 200, and 300. The training accuracies achieved by these models were impressive, with the final training accuracy for each model being approximately 99.22%, 99.22%, 99.69%, and 99.91%, respectively. These high training accuracies indicate that the models have learned to fit the training data very well.

Validation Accuracy:

However, the true test of the model's fitness lies in its validation accuracy, which reflects its ability to generalize beyond the training data. The validation accuracies for the four models were approximately 74.00%, 73.50%, 77.99%, and 78.00%, respectively. While the validation accuracies are decent, they are noticeably lower than the training accuracies, suggesting the presence of overfitting.

Addressing Overfitting:

To address the issue of overfitting, we employed several strategies:

1. Early Stopping: Early stopping was implemented with a patience of 3 epochs. This means that if the validation loss did not improve significantly for three consecutive epochs, the training process was halted. Early stopping proved effective as the models stopped training much earlier than the specified 50 epochs, preventing them from overfitting excessively.
2. Bidirectional LSTM: The model architecture includes Bidirectional Long Short-Term Memory (LSTM) layers. Bidirectional LSTMs allow the model to capture contextual information from both past and future tokens, aiding natural language processing tasks. This helped improve model performance and contributed to addressing overfitting.
3. Embedding Length: We experimented with different embedding lengths (50, 100, 200, and 300) to observe their impact on overfitting. Using larger embedding lengths, such as 300, potentially allows the model to learn more complex patterns in the data, thereby reducing overfitting.

Despite implementing strategies to mitigate overfitting, the models still exhibit signs of overfitting, as evidenced by the difference between training and validation accuracies. The early stopping criteria were effective in preventing extensive overfitting.

A screenshot of a graph

Description automatically generated3. Provide visualizations of the model’s training process, including a line graph of the loss and chosen evaluation metric.

4. Discuss the predictive accuracy of the trained network using the chosen evaluation metric from part D3.

The predictive accuracy of the trained network was evaluated using the chosen evaluation metric, which was the validation accuracy. Validation accuracy measures how well the model performs on new, unseen data, and it serves as a crucial indicator of the model's ability to generalize.

We trained several models with different word embedding lengths, specifically 50, 100, 200, and 300. The validation accuracy results for each model were as follows:

1. Model with an embedding length of 50: Validation accuracy of approximately 75.50%
2. Model with an embedding length of 100: Validation accuracy of approximately 74.00%
3. Model with an embedding length of 200: Validation accuracy of approximately 76.99%
4. Model with an embedding length of 300: Validation accuracy of approximately 73.00%

Interestingly, the model with an embedding length of 200 achieved the highest validation accuracy among all the models, indicating that it performed the best in predicting sentiment for movie reviews it had not seen during training. It is worth noting that all models showed signs of overfitting, as the training accuracy was consistently higher than the validation accuracy.

Part V: Summary and Recommendations

E. Provide the code you used to save the trained network within the neural network.

F. Discuss the functionality of your neural network, including the impact of the network architecture.

The neural network used for sentiment analysis on IMDb movie reviews demonstrates a powerful architecture designed to effectively classify movie reviews into positive or negative sentiments. It comprises three key components that play vital roles in its functionality:

1. **Embedding Layer**: The neural network begins with an embedding layer, which transforms each word in the input sequence into a dense vector representation. This step is essential as it allows the model to grasp the contextual meaning and semantic relationships between words. Consequently, words with similar meanings are represented closer together in the embedding space.
2. **Bidirectional LSTM Layer**: Following the embedding layer, the neural network incorporates a bidirectional Long Short-Term Memory (LSTM) layer. LSTMs are recurrent neural networks (RNNs) capable of capturing sequential dependencies in data. The bidirectional aspect further enhances the model by processing the input sequence in both forward and backward directions. This feature enables the model to consider information from both past and future words in the movie review, making it more proficient in understanding the overall sentiment expressed.
3. **Dense Layers**: The final part of the neural network consists of two dense layers. These layers handle the classification task, transforming the LSTM layer's output into a single output unit using a sigmoid activation function. As a result, the model can provide a probability score between 0 and 1, signifying the likelihood of the review being positive.

Functionality:

The neural network's functionality revolves around sequential processing of text, leveraging word embeddings, and employing bidirectional LSTM layers. The sequential processing aspect is crucial in handling textual data, allowing the model to understand the temporal context and dependencies present in movie reviews.

Word embeddings play a vital role in representing words in a continuous vector space, capturing their semantic meaning and relationships with other words. This enables the model to generalize better and comprehend the overall sentiment based on the context in which words appear.

The bidirectional LSTM layer is a key strength of the network. By considering both past and future words, the model gains a comprehensive understanding of the review's sentiment, making it adept at sentiment analysis tasks.

Impact of Network Architecture:

The neural network's architecture effectively handles sequential text data, optimizing its sentiment analysis performance. The incorporation of word embeddings and bidirectional LSTM layers significantly contributes to its success in understanding the context and meaning of movie reviews. By taking into account past and future words, the model can capture intricate dependencies, making it well-suited for natural language processing tasks.

G. Recommend a course of action based on your results.

1. **Movie Review Aggregation Platform:** We can build a user-friendly platform that aggregates movie reviews from various sources and leverages our NLP model to analyze the sentiment of those reviews. This platform will provide users with quick insights into the overall sentiment towards a movie, enabling them to make informed decisions about which movies to watch based on positive or negative feedback from other viewers.
2. **Content Curation and Recommendation:** Incorporating our sentiment analysis model into a content recommendation system, we can offer personalized movie recommendations to users based on their preferences and sentiments. By analyzing user-generated movie reviews, the system can identify movies that align with each user's tastes, leading to a more engaging and satisfying movie-watching experience.
3. **Marketing and Advertising Campaigns:** The NLP model's ability to analyze customer sentiment towards specific movies, genres, or actors can be harnessed for marketing and advertising campaigns. With the data insights from the sentiment analysis, movie studios and distributors can tailor their messaging and promotional efforts to resonate better with the target audience, thereby maximizing the impact of their campaigns and driving higher audience engagement.

Part VI: Reporting

H. Show your neural network in an industry-relevant interactive development environment (e.g., a Jupyter Notebook). Include a PDF or HTML document of your executed notebook presentation.

I. Denote specific web sources you used to acquire segments of third-party code that was used to support the application.

No sources were used.

J. Acknowledge sources, using in-text citations and references, for content that is quoted, paraphrased, or summarized.

No sources were used.

K. Demonstrate professional communication in the content and presentation of your submission.