Sentiment Analysis using Recurrent Neural Networks (RNNs) for Movie Reviews

By: Saurabh Chaudhary, Fall 2023, Cs-672

1. Introduction:

Sentiment analysis, also known as opinion mining, is the process of determining and extracting the sentiment or emotional tone expressed in a piece of text, such as a review, tweet, or comment. The goal is to understand whether the text conveys a positive or negative sentiment. In the context of movie reviews, sentiment analysis is particularly valuable as it helps gauge audience reactions, preferences, and overall reception of a film.

Understanding the sentiment behind movie reviews is crucial for several reasons:

Movie producers and studios can use sentiment analysis to understand how the audience perceives their films. Positive reviews can help in marketing and building anticipation for future releases, while negative reviews can point to areas that need improvement. Film critics play a significant role in shaping public opinion about a movie. Analyzing the sentiment of critic reviews can provide insights into the critical acclaim or backlash a film receives, influencing its reputation in the industry. Sentiment analysis is integral to building effective movie recommendation systems. By understanding user sentiments in reviews, recommendation algorithms can suggest movies that align with users' preferences. Studios and distributors can use sentiment analysis to tailor marketing strategies. Positive sentiments can be highlighted in promotional materials, while negative sentiments can inform strategies to address potential concerns or criticisms.

Recurrent Neural Networks (RNNs) are suitable for sentiment analysis tasks, especially in the context of movie reviews, due to their ability to capture sequential information in text data as they are designed to handle sequential data and can capture dependencies between words in a sentence. RNNs, with their memory of previous inputs, can capture contextual information and better understand the overall sentiment of a sentence. RNNs can process input sequences of different lengths, making them flexible for handling reviews of unusual sizes. And RNNs can capture long-term dependencies in sequences, allowing them to consider the entire context of a review. This is essential for understanding sentiments that may be expressed through a series of statements or paragraphs.

2. Data Preparation & Pre-requisite:

For this project I downloaded a csv file include 50k movie reviews from Kaggle & the link for the dataset is https://www.kaggle.com/datasets/lakshmi25npathi/imdb-dataset-of-50k-movie-reviews

To process and prepare the data I imported all the necessary libraries including numpy,pandas, re, nltk, sklearn, tensorflow & keras. I checked the shape and null values in the dataset then I created a new column to store the positive & negative sentiments as numbers, 0 for negative and 1 for positive sentiments. To remove all the stopwords, special characters and stem the words I created a method named stemming. For further processing I removed all the unnecessary columns for dataset. I used tarin test split to split the dataset into training and testing data.

3. Text Representation:

To deal with text and sentences I used tokenization technique which breaks down a text into tokens and ensures that all tokenized sequences are of the same length by adding padding to shorter sequences. Once tokenized, the sequences may have different lengths. To create a consistent input size for machine learning models, padding is applied to shorter sequences. Padding involves adding special tokens (usually zeros) to the shorter sequences until they match the length of the longest sequence in the dataset.

4. Model designing, Training & Evaluation:

The model is created as a sequential model. An embedding layer is added, representing a mapping of input indices to dense vectors of fixed size. The input vocabulary size is 500, and each word is embedded into a 16-dimensional vector. The input sequence length is set to 10. A Long Short-Term Memory (LSTM) layer with 32 units is added. The LSTM layer is designed to capture sequential dependencies in the input data. I chose LSTM over GRU because of the long-term dependency as the meaning of the review sentence may depend on words that are far apart.

Finally, a Dense layer with a single output neuron and a sigmoid activation function is added, indicating binary classification (0 or 1).

After designing the model, it is compiled using the Adam optimizer, binary cross-entropy loss function (suitable for binary classification), and accuracy as the evaluation metric. The model is trained using the training data padded with corresponding binary labels y_train. The training has been performed for 250 epochs with accuracy of 94.81%. By increasing the embedding dimensions from 16 to 32 the accuracy score is 87.43% on 100 epochs which is an improvement in our model accuracy. Then I added one more LSTM layer of 64 neurons & increased the embedding dimensions from 16 to 32. The model achieved 93.80% on 50 epochs.

The test data X_test is tokenized using the same tokenizer used for training. The sequences are padded to have a maximum length of 10. Predictions are made using the trained RNN model on the padded test data.

5. Conclusion:

In our project Long Short-Term Memory Recurrent Neural Networks (LSTM RNN) worked quite well with Accuracy and Precision of 62.75% and 62.2% respectively. By adding 2 layers of LSTM with 64 and 32 neurons and increasing the embedding dimension from 16 to 32 increased the model accuracy to 93.8% on 50 epochs which is very good compared to 80.85% accuracy on 50 epochs of our first model