CMPSC 448 Finals

Name: Yu Jie Yeo

Task: Sentiment Analysis using RNN and CNN

Abstract

This report presents a comparative analysis of Recurrent Neural Networks (RNNs) and Convolutional Neural Networks (CNNs) for sentiment analysis on IMDB movie reviews. The objective of this study is to understand the performance characteristics of RNNs and CNNs in the context of sentiment analysis and provide guidance on model selection for similar tasks. The methodology involves preprocessing the text data, training both RNN and CNN models on the preprocessed data and evaluating their performance. The key findings indicate that both models can effectively capture the sentiment of the text, with nuanced differences in performance.

Introduction

Sentiment analysis, also known as opinion mining, involves the use of natural language processing, text analysis, and computational linguistics to identify and extract subjective information from source materials. It is a powerful tool that is widely used in fields ranging from marketing to customer service to clinical psychology.

In this project, we aim to conduct a comparative analysis of two deep learning models, Recurrent Neural Networks (RNN) and Convolutional Neural Networks (CNN), for sentiment analysis on IMDB movie reviews. Both RNN and CNN have shown promising results in various natural language processing tasks, including sentiment analysis. However, each has its strengths and weaknesses, and their performance can vary depending on the task and data at hand.

RNNs are particularly suited for sequence prediction problems given their ability to handle sequential data by maintaining an internal state from time-step to time-step. On the other hand, CNNs, traditionally used in image processing, have been found to be effective for text-related tasks due to their ability to capture local and global patterns in data.

By comparing these two models on the same task, we hope to gain insights into their performance characteristics and provide guidance on model selection for sentiment analysis tasks.

Dataset Description

The dataset used for this project consists of movie reviews from IMDB. It contains 39,723 training samples, each labeled as either 1 or 0 representing positive and negative reviews, respectively.

Each review in the ‘text’ column represents a reviewer’s opinion about a particular movie. The ‘sentiment’ column is a binary classification, where 1 indicates a positive review and 0 indicates a negative review.

This dataset provides a rich resource for our task as it contains real-world, user-generated content with a wide range of sentiments, writing styles, and vocabulary. The binary classification simplifies our task to a two-class problem, allowing us to focus on the comparative analysis of the RNN and CNN models.

Link to the dataset can be found here.

Data Preprocessing

Data Distribution Analysis: We analyze the distribution of the dataset by checking how many positive and negative reviews there are respectively, which is an important step to understand the balance of classes in your dataset. A bar plot is used to visualize this distribution.

Text Cleaning and Normalization: A function `preprocess_text` was defined to clean and normalize the text data. This function performs several operations:

- Converts all text to lowercase, which helps to avoid duplication of words based on case differences.
- Replaces line breaks with spaces, which helps to maintain the continuity of sentences.
- Replaces three or more consecutive letters by two letters, which helps to handle elongated words.
- Replaces various types of emojis with corresponding text representations, which helps to capture the sentiment expressed by these emojis.
- Replaces hashtags with a special `<hashtag>` token followed by the actual hashtag text, which helps to separate the hashtag symbol from the hashtag text.
- Replaces contractions with their expanded forms using a contractions dictionary, which helps to standardize the text.
- Removes non-alphanumeric characters, which helps to focus on the actual words in the text.
- Replaces forward slashes with spaces, which helps to separate words that might be joined by slashes.
- Removes extra spaces, which helps to clean up the text.

Stopword Removal and Lemmatization: After cleaning and normalizing the text, I applied further preprocessing to each review. I removed stopwords, which are common words that do not contain important meaning and are thus often excluded from processing. I also applied lemmatization, which is the process of reducing inflected words to their base or root form. Both these steps help to focus on the important words in the text, and I can concentrate on the words that carry the most meaning.

Data Splitting: Finally, the data was split into training and test sets using an 80-20 split.

These preprocessing steps are crucial to prepare the text data for modelling as it helps to clean and standardize the text, reduce noise, and transform the text into a format that the model can understand.

Model Implementation

Recurrent Neural Network (RNN)

The RNN model used in this task is a simple one-layer RNN followed by two fully connected layers. The architecture of the RNN model is as follows:

Embedding Layer: This layer transforms the integer-encoded vocabulary to dense vector embeddings and uses pre-trained embeddings from the Word2Vec model.

SimpleRNN Layer: This is the recurrent layer of the model where temporal dependencies between words in the reviews are captured. It has 100 units and uses the ReLU activation function.

Dense Layer: This is a fully connected layer with 32 units and a ReLU activation function. It serves to interpret the features extracted by the RNN layer.

Output Layer: This is the final fully connected layer that outputs the prediction of the model. It has 1 unit and uses the sigmoid activation function, which is suitable for binary classification.

The model architecture is: Embedding -> SimpleRNN -> Dense -> Dense. The first Dense layer and the second Dense layer are the two fully connected layers aforementioned. The second Dense layer is also the output layer of the model. The model is compiled with the Adam optimizer and binary cross-entropy loss, which is suitable for a binary classification problem. The model is then trained using the training data with a validation split of 10% for monitoring the validation loss and accuracy during training. An EarlyStopping callback is used to prevent overfitting by stopping the training when the validation accuracy stops improving for a certain number of epochs. The min_delta parameter is set to 1e-4, meaning that an improvement is only considered as an improvement if it’s more than 1e-4. The patience parameter is set to 5, meaning that training will be stopped if there’s no improvement in the validation accuracy for 5 consecutive epochs. The model is then used to make predictions on the test data, and its performance is then printed out.

Convolutional Neural Network (CNN)

The CNN model used in this task is a simple one-layer CNN followed by two fully connected layers. The architecture of the CNN model is as follows:

Embedding Layer: The first layer is an Embedding layer, which transforms the integer-encoded vocabulary into dense vector embeddings. The model uses pre-trained embeddings from the Word2Vec model. The output of this layer is a 3D tensor of shape (batch_size, sequence_length, embedding_dimensions).

Conv1D Layer: The next layer is a 1D convolutional layer (Conv1D), which can be thought of as a sliding window that scans across the sentences. This layer has 100 filters and uses a kernel size of 5. Each filter learns to recognize a specific pattern across the input data. The activation function is ReLU (Rectified Linear Unit), which introduces non-linearity into the model. The output of this layer is a 3D tensor of shape (batch_size, steps, filters).

GlobalMaxPooling1D Layer: Following the Conv1D layer, a GlobalMaxPooling1D layer is used, which down-samples the input by taking the maximum value over the time dimension (steps). This operation extracts the most important features from the previous layer. The output of this layer is a 2D tensor of shape (batch_size, filters).

Dense Layer: The output from the GlobalMaxPooling1D layer is then passed through a Dense layer, also known as a fully connected layer. This layer has 32 units and uses a ReLU activation function. It serves to interpret the features extracted by the previous layers and passes them through the non-linear ReLU function.

Output Layer: The final layer is another Dense layer with a single unit and a sigmoid activation function. The sigmoid function squashes the output values between 0 and 1, providing the final probability for the positive class.

The model is compiled with the Adam optimizer and binary cross-entropy loss, suitable for binary classification tasks. During training, the model learns to map the input sequences to the correct sentiment labels (positive or negative). The EarlyStopping callback monitors the validation accuracy during training and stops the training process if the validation accuracy does not improve by at least 1e-4 for 5 consecutive epochs, which helps to prevent overfitting. The model is trained with a batch size of 1024 for up to 10 epochs, with 10% of the training data held out for validation. The model is then used to make predictions on the test data, and its performance is then printed out.

Training Details

Both models were trained using the following parameters:

Loss Function: Binary Cross-Entropy was used as the loss function. This is a suitable choice for binary classification problems.

Optimizer: The Adam optimizer was used. Adam is a popular choice due to its adaptive learning rate.

Batch Size: The batch size was set to 1024. This is the number of samples that are passed through the network at once. The choice of batch size can affect the speed and stability of the learning process.

Number of Epochs: The models were trained for up to 10 epochs, but training would stop early if the validation accuracy did not improve by at least 1e-4 for 5 consecutive epochs. This is controlled by the EarlyStopping callback. However, the models might have performed better if more epochs were added, but I used 10 only due to computational resources limitations.

Validation Split: A portion (10%) of the training data was set aside for validation during training. This helps to monitor the model's performance on unseen data and control overfitting.

Callbacks: An EarlyStopping callback was used to prevent overfitting. This callback stops training when the validation accuracy stops improving for a certain number of epochs (patience=5). The min_delta parameter is set to 1e-4, meaning that an improvement is only considered as an improvement if it's more than 1e-4.- Mention any techniques used to prevent overfitting, such as dropout or regularization.

Results

The performance of both the Recurrent Neural Network (RNN) and Convolutional Neural Network (CNN) models was evaluated using appropriate metrics such as accuracy, precision, recall, and F1-score.

Recurrent Neural Network (RNN) Model

The RNN model achieved an accuracy of 85.86% and an F1 score of 86.53%. The precision and recall for the negative class were 0.89 and 0.82 respectively, and for the positive class were 0.83 and 0.90 respectively. This suggests that the RNN model was slightly better at identifying positive reviews than negative reviews.

For the RNN model, the confusion matrix is:

[3116 850]

[366 3668]

This indicates that the RNN model correctly classified 3116 negative reviews (true negatives) and 3668 positive reviews (true positives). However, it incorrectly classified 850 negative reviews as positive (false positives) and 366 positive reviews as negative (false negatives).

Convolutional Neural Network (CNN) Model

The CNN model achieved an accuracy of 86.42% and an F1 score of 86.50%. The precision and recall for the negative class were 0.86 and 0.87 respectively, and for the positive class were 0.87 and 0.86 respectively. This suggests that the CNN model performed similarly on both positive and negative reviews.

For the CNN model, the confusion matrix is:

[3270 696]

[423 3611]

This indicates that the CNN model correctly classified 3270 negative reviews (true negatives) and 3611 positive reviews (true positives). However, it incorrectly classified 696 negative reviews as positive (false positives) and 423 positive reviews as negative (false negatives).

Observations

Both models achieved similar performance on the test data, with the CNN model performing slightly better in terms of overall accuracy. The performance of both models suggests that they were able to effectively capture the sentiment of the movie reviews.

From the confusion matrices, we can observe that both models have more false positives than false negatives. This means that they are more likely to incorrectly classify a negative review as positive than to incorrectly classify a positive review as negative.

The CNN model has fewer false positives and more true negatives than the RNN model, indicating that it is slightly better at correctly classifying negative reviews. On the other hand, the RNN model has fewer false negatives and more true positives, indicating that it is slightly better at correctly classifying positive reviews.

The training process of both models showed steady improvement in accuracy over epochs, indicating that the models were learning effectively from the training data. The use of early stopping helped prevent overfitting by stopping the training process when the validation accuracy stopped improving.

Conclusion

In conclusion, both RNN and CNN models can be effectively used for sentiment analysis tasks. The choice between RNN and CNN may depend on the specific requirements of the task and the nature of the data. In this case, both models achieved similar performance, but the CNN model was slightly more accurate.

The performance of these models could potentially be improved with further tuning of the hyperparameters, use of more complex model architectures, or use of additional techniques to prevent overfitting. However, the results obtained in this study provide a good starting point for further exploration of this topic.

RNNs, particularly their advanced variant LSTMs, are often favored for sentiment analysis due to their proficiency with sequential data like text. In this study, a simple RNN was used, which lacks the ability to remember long-term dependencies, a problem addressed by LSTMs. Therefore, using an LSTM could potentially improve performance. While the RNN and CNN models used in this study yielded satisfactory results, exploring more complex RNN variants like LSTMs could lead to enhanced performance.

Finally, this study demonstrates the power of deep learning techniques for natural language processing tasks and provides a practical example of how these techniques can be applied to sentiment analysis. It also highlights the importance of data preprocessing and model evaluation in the machine learning workflow.

Challenges & Solutions

Given the open-ended nature of the project, selecting a specific topic and dataset was initially challenging. After careful consideration, I chose to focus on sentiment analysis of movie reviews, a topic that piqued my interest and had ample references available online due to its popularity in the field.

Understanding the Model Architecture: One of the initial challenges was comprehending the architecture of the RNN and CNN models and understanding the role each layer plays in the overall performance of the model. To overcome this, I referred to lecture notes, consulted various online sources such as StackOverflow, and read up on the documentation for RNN and CNN implementation.

Data Preprocessing: Data preprocessing often poses a significant challenge in machine learning projects due to the myriad ways it can be approached, and it often boils down to trial and error, heavily dependent on the nature of the dataset. It was enlightening to explore various preprocessing methods employed by others on Kaggle. Drawing upon their knowledge, I followed some best practices for preprocessing.

Interpreting the Results: Interpreting the results of the models was indeed a challenge. To tackle this, I utilized data visualization libraries to plot the outcomes of my models, which provided a clearer visual representation of their performance. Additionally, I generated a classification report and a confusion matrix for each model. These tools gave me a comprehensive view of the models' performance across various metrics, including precision, recall, and F1 score. Additionally, I referred to my lecture notes and conducted additional research to grasp the significance of each metric. This enabled me to correlate the results back to the performance of my models effectively. By doing so, I was able to gain valuable insights into the strengths and weaknesses of each model, which is crucial for refining the models and improving their performance in future iterations.

Improving Model Performance: Identifying ways to enhance the performance of the models was another challenge, given the multitude of parameters that could be tuned and the trade-offs to consider, such as between accuracy and computational resources. I endeavoured to test various methods and fine-tune hyperparameters to achieve optimal results. However, the most crucial aspect was ensuring the experiment was conducted correctly as a whole. Given more time and resources, I believe I could attain better results, but I am content with the outcomes achieved in this project.

References

Bajaj, T. (n.d.). Movie Reviews Sentiment Analysis. Kaggle. Retrieved from https://www.kaggle.com/code/tanavbajaj/movie-reviews-sentiment-analysis/notebook.

Shinde, S. (n.d.). Multi-Channel CNN Model for Sentiment Analysis. Kaggle. Retrieved from https://www.kaggle.com/code/shivamshinde123/multi-channel-cnn-model-for-sentiment-analysis.