COMP 562 Introduction to Machine Learning

Fall 2024

Final Project Report

Project Title: Sentiment Analysis with RNN and CNN

Group Details:

First Name, Last Name	PID
Jun Xian Marcel, Lee	730826716
Xing Hui Bertrand, Wong	730828121
Roderich Suwandi, Lee	730827236
Mikhil, Anand	730827784

1. 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 four deep learning models, Recurrent Neural Networks (RNN), Convolutional Neural Networks (CNN), CNN + Long Term Short Memory (LTSM) and BERT for sentiment analysis on IMDB movie reviews. All models 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. BERT is based on the Transformer architecture, which relies on self-attention mechanisms to allow every word in the sentence to attend to every other word, regardless of their distance in the text. However, it is more computationally expensive than CNN or RNN models.

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

3. 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 models.

Link to the dataset can be found here.

4. Data Preprocessing

Data Distribution Analysis: We first 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 such as changing all text to lowercase, replace line break with spaces and remove/substitute special characters with normal letters. We further preprocess each review by removing stopwords and apply lemmatization in order to concentrate on the words that carry the most meaning. 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.

5. Model Implementation

Convolutional Neural Network (CNN)

The CNN model for this task consists of a one-layer Conv1D followed by two fully connected layers. Its architecture is as follows:

- **Embedding Layer**: Converts integer-encoded vocabulary into dense vector embeddings using pre-trained Word2Vec, outputting a 3D tensor of shape `(batch_size, sequence_length, embedding_dimensions)`.
- **Conv1D Layer**: A 1D convolutional layer with 100 filters, a kernel size of 5, and ReLU activation scans across sentences, learning patterns. It outputs a 3D tensor of shape `(batch_size, steps, filters)`.
- **GlobalMaxPooling1D Layer**: Down-samples the Conv1D output by taking the maximum value over the time dimension, reducing it to a 2D tensor of shape `(batch_size, filters)`.
- **Dense Layer**: A fully connected layer with 32 units and ReLU activation interprets extracted features.
- **Output Layer**: A Dense layer with 1 unit and sigmoid activation provides the final probability for binary classification.

Recurrent Neural Network (RNN)

The RNN model for this task consists of a one-layer SimpleRNN followed by two fully connected layers. It uses the following architecture:

- **Embedding Layer**: Maps integer-encoded vocabulary to dense vector embeddings using pre-trained Word2Vec.
- **SimpleRNN Layer**: Captures temporal dependencies with 100 units and ReLU activation.
- **Dense Layers**: The first dense layer (32 units, ReLU) interprets RNN features, while the second dense layer (1 unit, sigmoid) outputs binary classification predictions.

CNN + Long Term Short Memory

The addition of a single LSTM layer with 64 units after CNN layers captures temporal dependencies in the sequential data, with tanh activation. By combining the two models, CNN + LSTM can first extract the key n-grams with CNN, reducing the dimensionality of the input data, and then feed these features into the LSTM to get a deeper understanding of the sentiment over the entire text. This hybrid approach allows the model to focus both on specific word combinations (n-grams) and overall context.

Example: In the sentence "The movie was good, though it was a bit slow at times," the CNN can capture the key n-grams like "good" and "bit slow," while the LSTM can determine that despite the negative connotation of "slow," the overall sentiment is positive due to the strong "good."

DistilBERT

The DistilBERT model is used for text classification, leveraging a pre-trained transformer for binary sentiment analysis. The workflow begins with preprocessing the dataset, where text reviews are tokenized using the DistilBERT tokenizer. Tokenized sequences are truncated or

padded to a maximum length of 128 and converted into TensorFlow tensors alongside their corresponding labels. The data is organized into TensorFlow datasets for efficient training and testing, with a batch size of 32 and shuffling applied to the training data.
The model architecture, TFDistilBertForSequenceClassification, is pre-trained on the "distilbert-base-uncased" variant and fine-tuned for binary classification. An optimizer is created with a learning rate of 5e-5, weight decay, and no warmup steps, designed for 5 epochs. It is then evaluated on tests data, and its performance is reported.

6. Training Details

Models were trained with binary cross-entropy loss and the Adam optimizer. The batch size was 1024, and training was limited to 10 epochs, with EarlyStopping halting training if validation accuracy did not improve by more than 1e-4 for 5 consecutive epochs. A 10% validation split was used to monitor performance and prevent overfitting. EarlyStopping and the validation split helped mitigate overfitting, but computational resource limitations restricted training to 10 epochs.

7. Results

The performance of the models was evaluated using appropriate metrics such as accuracy, precision, recall, and F1-score.

Convolutional Neural Network (CNN) Model

The CNN model achieved an accuracy of 86.20% and an F1 score of 86.37% with confusion matrix:
[3398 568]
[536 3498]
This indicates that the CNN model correctly classified 3398 negative reviews (true negatives) and 3498 positive reviews (true positives). However, it incorrectly classified 568 negative reviews as positive (false positives) and 536 positive reviews as negative (false negatives).

Recurrent Neural Network (RNN) Model

The RNN model achieved an accuracy of 85.52% and an F1 score of 86.58% with confusion matrix:
[3108 858]
[300 3734]
This indicates that the RNN model correctly classified 3108 negative reviews (true negatives) and 3734 positive reviews (true positives). However, it incorrectly classified 858 negative reviews as positive (false positives) and 300 positive reviews as negative (false negatives).

Convolutional Neural Network (CNN) Model with Long Term Short Memory (LTSM)

The CNN + LTSM model achieved an accuracy of 85.25% and an F1 score of 86.73% with confusion matrix:
[2964 1002]
[178 3856]
This indicates that the CNN model correctly classified 2964 negative reviews (true negatives) and 3856 positive reviews (true positives). However, it incorrectly classified 1002 negative reviews as positive (false positives) and 178 positive reviews as negative (false negatives).

BERT (Distil Version) Model

The BERT model achieved an accuracy of 88.95% and an F1 score of 89.04% with confusion matrix:

[3524 442]

[442 3592]

This indicates that the CNN model correctly classified 3524 negative reviews (true negatives) and 3592 positive reviews (true positives). However, it incorrectly classified 442 negative reviews as positive (false positives) and 442 positive reviews as negative (false negatives).

8. Observations

All models effectively captured the sentiment of movie reviews, with BERT achieving the highest accuracy (88.95%). The CNN and BERT models had around the same proportion of false positives and false negatives, while the RNN and CNN + LSTM had a larger proportion of false positives to false negatives.

9. Conclusion

Our results indicate that all models effectively captured the sentiment of movie reviews (ranging from 85%-89% accuracy), with BERT (DistilBERT version) outperforming the other models in terms of accuracy (88.95%) and F1 score (89.04%). However, while BERT demonstrated superior performance, it came at the cost of increased computational complexity, making it more resource-intensive compared to the other models.

The CNN model achieved a respectable accuracy of 86.20% with an F1 score of 86.37%, making it a solid choice for sentiment analysis tasks where efficiency is important. The simple RNN model also performed well, achieving an accuracy of 85.52% and an F1 score of 86.58%, with a slight advantage in capturing positive reviews, as indicated by its confusion matrix. This shows that the simple RNN is also a solid choice for sentiment analysis tasks, considering its efficiency (as it is simple) and its ability to understand sequential data. We tried to improve upon both CNN and RNN models by combining them together into a CNN + LSTM model, but it did not show significant improvements over the individual CNN and RNN models. This result might suggest that accuracy in sentiment analysis is not easily enhanced by simply combining two simpler models. It could imply that the IMDB review sentimental analysis problem requires more sophisticated approaches or architectures to capture the complexity of the task more effectively. This idea is supported by the fact that a simple BERT model was able to outperform all CNN and RNN models, because BERT's attention mechanism enables it to model relationships between words that are far apart in a sentence or paragraph, which is important for sentiment analysis.

Moreover, the CNN and BERT models had around the same proportion of false positives and false negatives, while the RNN and CNN + LSTM had a larger proportion of false positives to false negatives. This distinction is important in certain applications, such as IMDB review sentiment analysis. For instance, if a system misclassifies too many negative reviews as positive (false positives), it might lead to misleading insights, such as overestimating audience satisfaction with a poorly received movie.

In conclusion, depending on the available computational resources and the specific requirements of the task, each model has its own merits. BERT is ideal for scenarios where accuracy is the top priority, and computational resources are not a limiting factor. In contrast, CNN and RNN provide viable alternatives when faster and less resource-demanding solutions are needed without compromising too much on performance.