Text Mining and Natural Language Processing

2022-2023

[Luca Ponzini 506276]

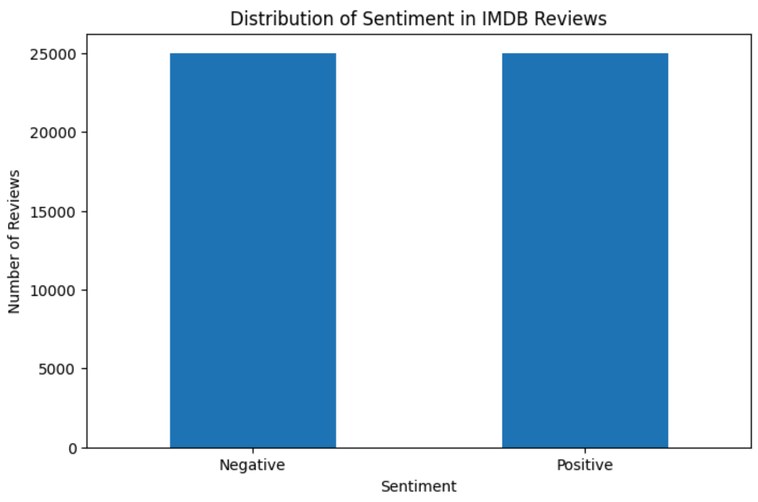
[Positive_or_Negative]

Introduction

This project leverages Natural Language Processing (NLP) techniques to classify sentiments in textual data. It encompasses preprocessing steps like tokenization and numerical conversion of labels, followed by the deployment of NLP models for sentiment classification. The aim is to accurately discern positive or negative sentiments within reviews of movies, showcasing the application of NLP in understanding human emotions expressed through language.

Data

The dataset I used is the IMDB Dataset - Sentiment Analysis. It is structured into 2 columns: “review” and “sentiment”: the first contains the text of the movie reviews, while the other column contains the “sentiment” associated to the meaning of each review, which can either be positive or negative.
In this dataset 50000 reviews are stored, regarding movies on the IMDB website, and among those, 25k are labeled as positive and 25k are labeled as negative, providing a very well balanced dataset for the analysis.



The dataset contains a total of 11557847 words across all reviews, in average each review is composed of 231 words, with the longest being formed by 2470 words and the shortest by 4.
There are no missing values in the dataset, meaning that for each review there is a corresponding “sentiment” value.
The presence of both short and long reviews allows for testing the models' effectiveness across different levels of textual complexity and information density.

Methodology

In my project, I embarked on a comprehensive journey to analyze sentiment in IMDB movie reviews, with an approach that begins from data preprocessing and that has the aim to analyze machine learning models for prediction.

Data Preprocessing

- Removing Special Characters:

  I began by cleansing the text data, stripping away all non-alphanumeric characters from the reviews. This step was pivotal in standardizing the dataset, ensuring consistency across all entries.
- Case Normalization:

  To further homogenize the data, I converted all text to lowercase. This crucial step ensures that words are uniformly recognized, regardless of their casing in the original text.
- Tokenization:

  Utilizing TensorFlow's `SubwordTextEncoder`, I tokenized the dataset into subwords.
  This approach not only optimizes the vocabulary for my specific dataset but also elegantly handles out-of-vocabulary words by breaking them down into recognizable subwords.
  I selected a vocabulary size and applied this tokenizer across all reviews, translating them into sequences of integers that represent each subword.
- Sequence Padding:

  To align the data input for model training, I padded these sequences to a uniform length. This ensures that every input fed into the models maintains a consistent dimension, facilitating batch processing and model efficiency.

Building and Comparing Models

In my exploration, I constructed and evaluated five distinct neural network architectures:

1. LSTM: I utilized LSTM networks to capture temporal dependencies, recognizing their process in learning order importance in sequence prediction.

2. GRU: I experimented with GRU networks as a streamlined alternative to LSTM, aiming to achieve similar temporal comprehension with potentially faster training times.

3. Conv1D: My methodology included Conv1D networks to extract features from the sequential data, leveraging their NLP capabilities.

4. SimpleRNN: I explored SimpleRNNs to understand their capability in capturing temporal patterns, despite their simplicity and known limitations compared to more advanced RNNs.

5. Transformer: I adopted Transformer models for their advanced self-attention mechanism, enabling each word to be processed in the context of the entire sequence, thus potentially enhancing contextual understanding.

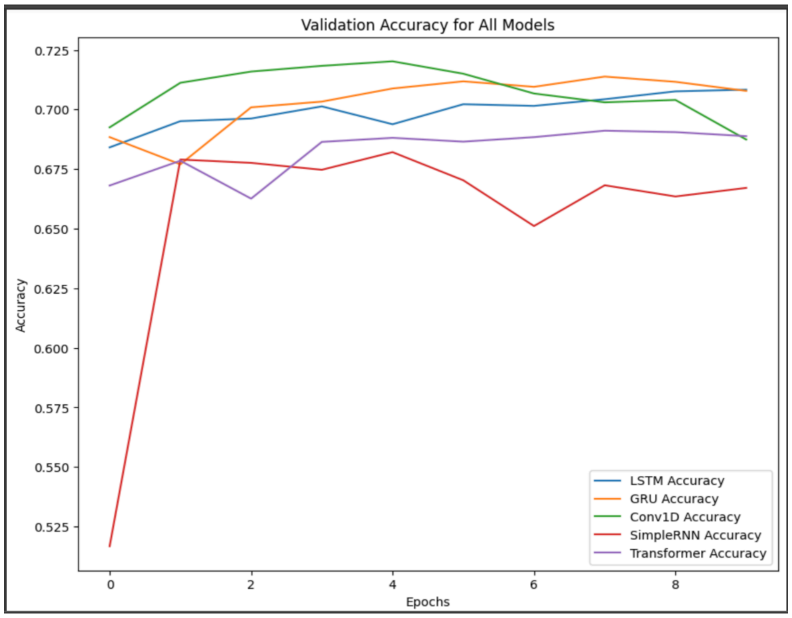
Each model begins with an embedding layer, transitioning into architecture-specific layers designed to capture the nuances of sentiment in the reviews. I opted for a binary cross-entropy loss function, aligning with the binary nature of the sentiment analysis task.

Training and Evaluation

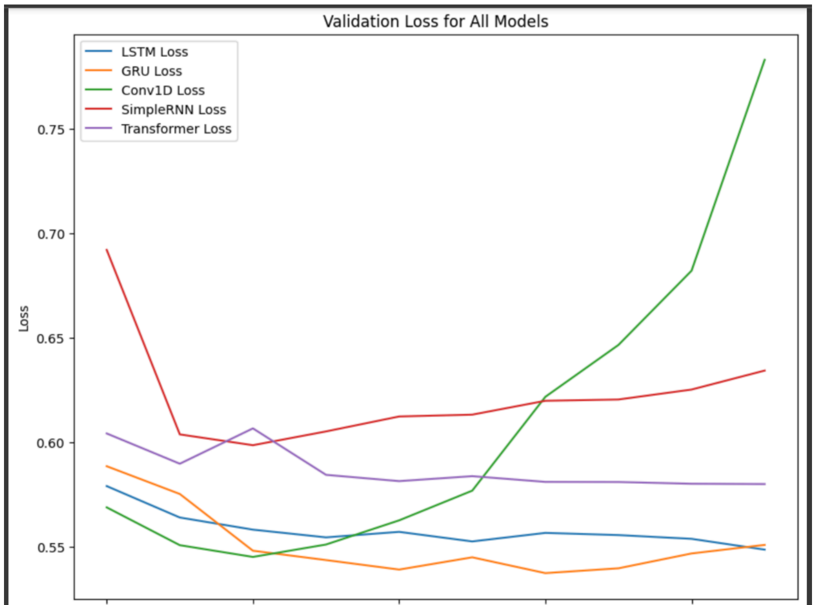
I divided the dataset, allocating 80% for training and the remainder for testing, to teach the models to differentiate between positive and negative sentiments accurately.
I then compared the models' accuracy and visualized their efficacy using confusion matrices, allowing me to point out the most effective architecture for sentiment analysis.
After understanding which model works better I then applied it on new reviews. This demonstration included preprocessing these reviews, predicting their sentiments, and ranking them by their predicted sentiment scores.
Throughout the process, I utilized various plots to illustrate sentiment distribution and model performance.
Incorporating diagrams of the model architectures could further enrich the presentation, offering a clearer understanding of each model's structure.

Results

After meticulously training and evaluating several models, I documented their performance to compare their efficacy.
The Conv1D model emerged as the top performer, showcasing its aptitude for capturing the sequential patterns and nuances within text data efficiently. Its architecture, designed to process data in one-dimensional sequences, is particularly well-suited for tasks like sentiment analysis where context and the order of words play a crucial role.
The GRU model also showed promising results, benefiting from its attention mechanism to focus on relevant parts of the input data when making predictions.



This graph visually compares the validation accuracy of the five different neural network models I used:
LSTM, GRU, Conv1D, SimpleRNN, and Transformer.
Over a number of epochs (10) during the model training process.



This graph depicts the loss of various neural network models on the validation set throughout the training epochs(10).Loss is a measure of how well the model is performing, with lower values indicating better performance.

Conclusion

The use of Conv1D for sentiment analysis proved to be highly effective, balancing computational efficiency with predictive performance. It reinforced my understanding of how convolutional layers can capture local dependencies in text data.
The initial preprocessing and the choice of subword tokenization were pivotal in preparing the dataset for neural network models, highlighting the importance of thoughtful data preparation in NLP tasks.
This project not only enhanced my skills in applying deep learning models to NLP problems but also provided valuable insights into the practical aspects of model selection, training, and evaluation in a real-world sentiment analysis task.