Sentiment Analysis on Movie Reviews

Project Report for Course Project: NLP DLBAIPNLP01

Mohsen Nasiri

Matriculation Number: 32206338

Tutor’s Name: Simon Martin

31.1.2025

Table of Contents

1. Introduction

This project focuses on developing a deep learning based Natural Language Processing (NLP) system to perform binary sentiment analysis on publicly available movie reviews. While traditional machine learning algorithms struggle with sequential dependencies and word context, Recurrent Neural Network (RNN) was chosen for its ability to capture bidirectional context.

The report will contain description of the system's development process, performance, and evaluation. The task specifically requires developing a binary classification model to determine whether a review expresses positive or negative sentiment, thus neutral reviews or values have been excluded from the project.

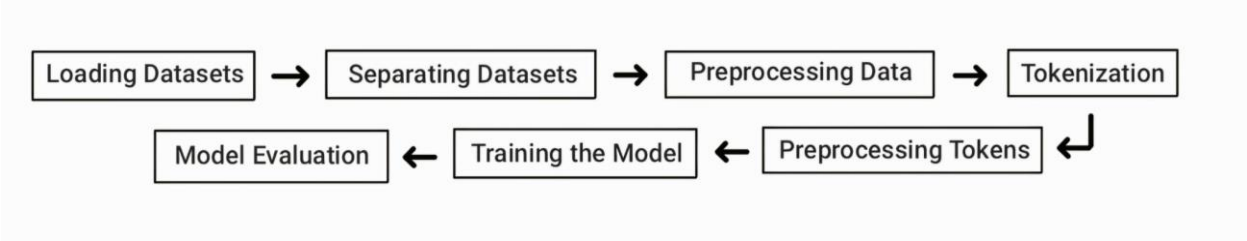
1.1. Objectives

The system is designed to:

- Collect movie review data from publicly available datasets (e.g., Stanford Sentiment Treebank and IMDB).
- Preprocess the data using NLP techniques, including:
- Train a binary classification model using deep learning with a bidirectional LSTM network to predict sentiment labels (positive or negative).
- Evaluate model performance using metrics such as accuracy, precision, recall, F1-score, and confusion matrices.
- Provide an interactive feature to classify user-input movie reviews, allowing real-time sentiment prediction.

2. Methodology

The project is designed to be modular, and can be divided into four main phases, data preparation, tokenization, model development, and model evaluation.



2.1. Data Preparation

The first phase involves preparing the datasets for training and validation. Datasets are downloaded directly inside the program using the Pandas library from Hugging Face and stored in a structured directory. If the datasets already exist locally, they are reused to save time. The preparation process includes the following key steps:

- **Column Selection:** Irrelevant columns are removed, and the remaining columns are renamed, leaving one for reviews and one for ratings.
- **Duplicate Removal:** Any duplicate rows in the datasets are identified and removed to ensure the data is clean and consistent.
- **Rating Standardization:** The rating column is transformed and unified to represent 0 for negative and 1 for positive.
- **Text Preprocessing:** Reviews are cleaned and processed in batches, using the NLP pipeline. This includes lemmatization, removal of HTML tags, and filtering of customized stopwords.
- **Dataset Combination:** The datasets are cleaned of empty rows and combined into a single, larger dataset.

After removing duplicates, cleaning, and preprocessing the datasets, they were combined, and the distribution of positive and negative sentiments was balanced. Finally, a dataset containing 136,000 reviews was split into a validation set with 5,000 reviews and a training set with 131,000 reviews. In general, increasing the number of unique reviews used for training improves the model's accuracy and performance. Therefore, the validation set was kept smaller to supply more data for training.

2.2. Tokenization

Once the data is prepared, text reviews need to be converted into numerical representations for training. This involves creating a tokenizer, which maps words to numeric tokens based on their frequency and importance in the dataset. The key steps in this phase include:

- **Prioritizing Key Words:** Words that contain sentiment are separated using NLTK sentiment analyzer, as well as those that modify sentiments (e.g., "not," "never"), After separation they are prioritized to ensure they are included in the model's vocabulary.
- **Filtering Non-English Words:** All words are checked against multiple English dictionaries to exclude non-English or uncommon terms that do not contribute to sentiment analysis.
- **Removing Named Entities:** Named entities such as dates, countries, or human names are excluded, as they generally do not affect the sentiment or context of a sentence.

At the end of this phase, tokenized text and their corresponding labels are ready for model training.

2.3. Model Development

The model architecture is designed to handle text data and capture complex and simple patterns for sentiment analysis. The following components shape the model:

1. **Embedding Layer:** Converts tokenized inputs into dense vector representations that encode semantic relationships between words.
2. **SpatialDropout1D Layer:** Reduces overfitting, outperforming traditional dropout in this sequence.
3. **Bidirectional LSTM (Long Short-Term Memory) Layer:** Processes the input data in both forward and backward directions, allowing the model to capture contexts, such as understanding phrases like "not good" or "not bad."
4. **Dropout Layer:** Further improves generalization by preventing reliance on specific neurons.
5. **Dense Output Layer:** Classifies sentiments as positive or negative by utilizing sigmoid activation function.

RNNs are well suited for sequential data due to their memory cells, which captures context across time steps. However standard RNNs, suffer from vanishing gradients. LSTM addresses this issue using gating mechanisms. Bidirectional LSTM further enhances this by analyzing text in both forward and backward directions, capturing context from entire sentences.

2.4. Model Evaluation

Once the model is trained, its performance is evaluated, and real-time predictions can be made. Key aspects include:

- **Performance Metrics:** Metrics such as accuracy, precision, recall, and F1 score are calculated. A confusion matrix is visualized.
- **Error Analysis:** Incorrect predictions are analyzed to identify areas where the model struggles, providing insights for improvements.
- **Training History Visualization:** Trends in accuracy and loss during training and validation are visualized to assess model learning and detect overfitting.
- **User based Evaluation:** An interactive system allows the user to input movie reviews manually, which gets processed to match the structure of the training data. The program identifies words not encountered during training and highlights them for better

interpretability. The sentiment of the review is then predicted and displayed as either positive or negative.

3. Setup and Tools

To ensure library compatibility and prevent version conflicts, the project was developed using Python 3.10, which is required for TensorFlow 2.16, and in Anaconda virtual environment.

3.1. Environment

After installing Anaconda, the environment can be created by navigating to the project directory in the terminal and running the following commands:

```
conda env create -f environment.yml
```

```
conda activate Movie_Sentiment_Analysis
```

3.2. Libraries

To perform natural language processing, model creation, evaluation, and other supporting tasks. These following libraries were selected.

Natural Language Processing (NLP) Libraries:

- **NLTK:** Utilized during preprocessing tasks, such as removing stopwords using its stopwords list and accessing the WordNet database for word validation and detecting words that contain sentiment using “SentimentIntensityAnalyzer”.
- **SpaCy:** Used for more advanced NLP tasks, particularly named entity recognition (NER), to identify and remove entities like locations, dates, and other irrelevant elements in the context of sentiment analysis. Also used for tokenization and lemmatization to normalize textual data efficiently.
- **Regular Expressions (re):** Utilized for preprocessing tasks such as cleaning text, removing HTML tags, and filtering out unwanted characters in the reviews.
- **Enchant:** Implemented for its dictionary to validate and filter English words during preprocessing. This ensured that only meaningful and correctly spelled words were retained for analysis.

Deep Learning Libraries:

- **TensorFlow 2.16:** Used as the core deep learning framework. At the time, this version was most compatible with Python 3.10 and the other libraries implemented, ensuring the least issues and errors.

- **Keras (a high-level API of TensorFlow):** Used to design and train the RNN using the Sequential framework, which offers a linear stack of layers. The Tokenizer is employed for the word list, and the Nadam optimizer is used to optimize the model during training.
- **Keras Tuner:** Implemented to find the best hyperparameters and optimize the number of layers inside the model, enhancing its performance and accuracy. It is not included in the final code since the optimization was done during drafts and tests.

Evaluation Libraries:

- **Scikit Learn:** Employed for calculating the evaluation metrics, including accuracy, precision, recall, and F1-score, to measure the model's performance.
- **Seaborn and Matplotlib:** Implemented to create heatmap visualizations for the confusion matrices.

Supporting Libraries:

- **Pandas:** Utilized for downloading, loading, cleaning, and manipulating datasets.
- **NumPy:** Used for array operations and for manipulating training data arrays.
- **Pickle:** Used to save and load the trained tokenizer, facilitating reproducibility and faster deployment.
- **JSON:** Utilized for saving and loading additional variables specifically the training history.
- **Operating System (os):** Employed to verify the existence of datasets and files, as well as manage directories.

3.3. Dataset

The following datasets were combined to identify broader sentiment patterns and enhance the analysis.

- **Stanford Sentiment Treebank (SST-2)** is a binary sentiment classification dataset with nearly 70,000 reviews from the SST project (Socher et al., 2013). While SST-2 provides labels for positive and negative sentiments only, a more comprehensive understanding of sentiment could have been achieved with SST-5, which includes five sentiment categories, or with a dataset incorporating neutral sentiments. However, binary sentiment classification was specified for the task thus SST-2 was utilized.
- **Stanford IMDB** is based on user reviews from the IMDB website and contains 50,000 reviews with binary sentiment labels, same as the SST-2 dataset (Maas et al., 2011). This dataset provides a more concentrated movie critics pattern rather than sentiment.

- **Rotton Tomatoes** is a collection of movie reviews extracted from the Rotten Tomatoes website, and it contains more than 10,000 reviews labeled with binary sentiments (Pang & Lee, 2005).
- **Arize AI Movie Reviews with Context Drift** was designed for use in a tutorial on dataset drift analysis. It is based on a large movie review dataset, with additional samples from a hotel review dataset (Arize AI, n.d.). The dataset consists of 12,500 reviews.
- **IMDB Rating Dataset** was utilized, and it was gathered by a user on Kaggle that contains approximately 7,000 reviews (itsabba3, n.d.). However, this dataset included user reviews along with movie ratings ranging from 1 to 10. To make it suitable for sentiment analysis, ratings in the range of 1 to 3 were considered negative, while those in the range of 8 to 10 were considered positive. The purpose of using this dataset was to experiment with datasets that lack predefined sentiment labels to simulate working with scraped data.

4. Results

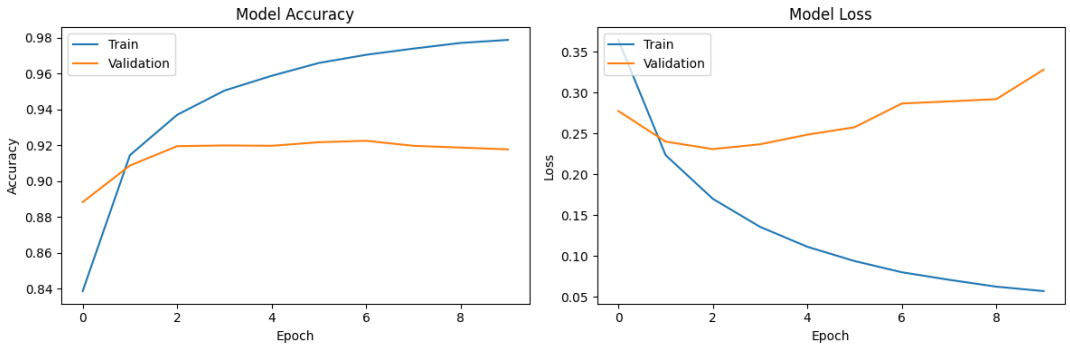
The result of this project consists of the evaluation of the model's performance, comparisons of different configurations, and the challenges encountered during development.

4.1. Performance

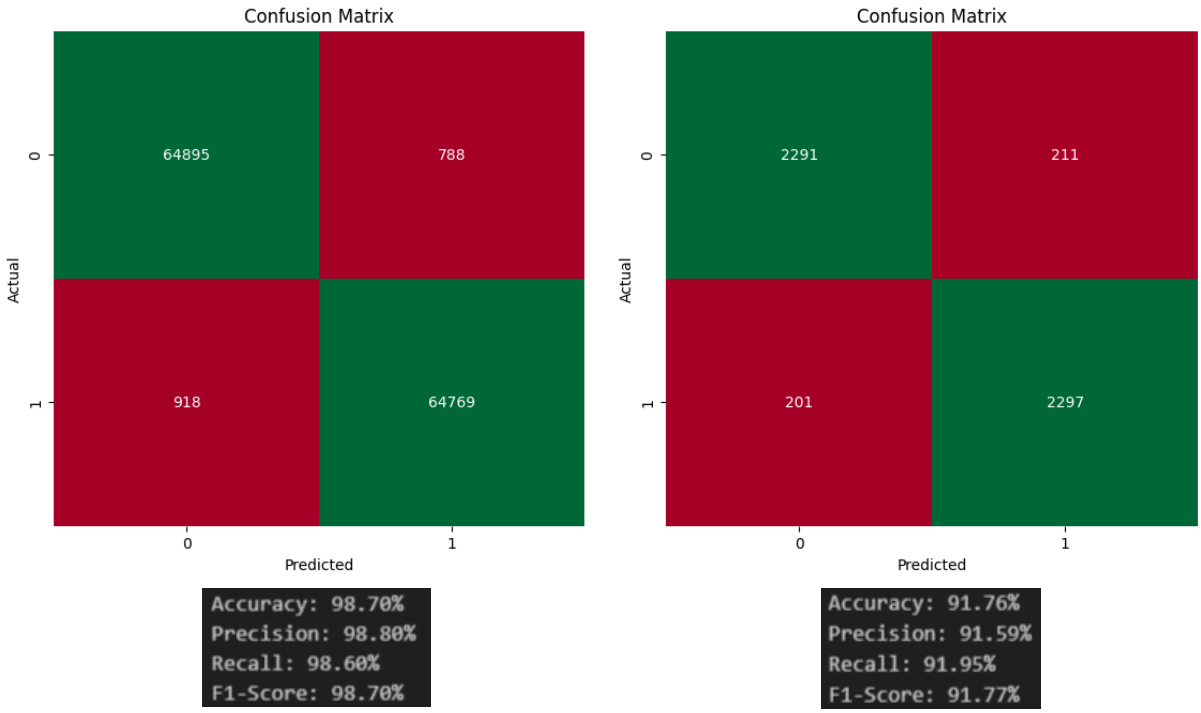
The following figures present the evaluations and results, including accuracy, loss, and a confusion matrix for the training and validation dataset.

Layer (type)	Output Shape	Param #
embedding (Embedding)	(None, 100, 128)	2,560,000
spatial_dropout1d (SpatialDropout1D)	(None, 100, 128)	0
bidirectional (Bidirectional)	(None, 256)	263,168
dropout (Dropout)	(None, 256)	0
dense (Dense)	(None, 1)	257

Model's Summary



Model's History



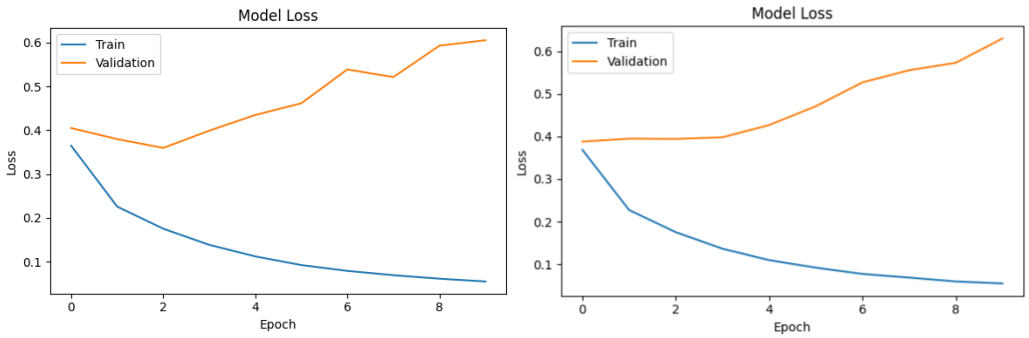
Model's Performance on Training (left) and Validation (right) Datasets

4.2. Comparison

Several approaches and configurations were evaluated during the development of the model. Comparisons were made across optimizers, model architectures, and preprocessing strategies to determine the most effective setup. Below are the key findings:

Optimizers

- The Nadam (Nesterov-accelerated Adaptive Moment Estimation) optimizer proved itself as the best choice, outperforming both RMSprop (Root Mean Square Propagation) and Adam (Adaptive Moment Estimation) in terms of model accuracy and stability during training. Nadam's adaptive learning rate helped achieve faster convergence compared to the other optimizers.



Model's Training History Using Adam Optimizer (left) and RMSprop Optimizer (right)

Model Architecture

- LSTM layers demonstrated superior performance compared to the simpler GRU layers. LSTMs were better at capturing long-term dependencies in the text, which is critical for sentiment analysis.
- Bidirectional LSTM layers provided significant improvements, especially for handling sentences such as "not good" or "not bad." This configuration enabled the model to analyze context from both directions, improving sentiment classification accuracy.
- Adding extra LSTM layers to enhance the model's depth resulted in minimal or no improvement in performance at the cost of increased training time.
- Batch normalization did not lead to improvements in performance but rather added complexity without benefits. Therefore, it was unnecessary for this architecture and dataset.

Regularization Methods

- L1 and L2 regularization and early stopping was employed to avoid overfitting. However, early stopping tended to terminate training early, not allowing the model to learn effectively. And in many cases, the model proved worthy of performance. In the end, these regularization methods were unnecessary and instead dropout layers were implemented.

Preprocessing Strategies

- Initially, preprocessing was performed on the entire data during the datasets loading phase, which proved to be highly time consuming. Switching parts of the text processing function such as the named entity removal and word validation to the tokenization phase, significantly reduced the computational overhead while maintaining the same effects.

4.3. Challenges

Designing the model architecture and tuning the parameters was complex. This issue was tackled with the help of Keras Tuner, which automated the process of identifying the optimal number of layers, their positions, and their hyperparameters. Specifically, it helped determine the appropriate number of dropout layers and LSTM layers, as well as selecting the best learning rate and dropout values.

Batch Size and Epochs

Finding the right amount for batch size and epochs required a lot of trial and error. Initially, larger batch sizes and higher epochs were used, which led to inconsistent training.

Handling Neutral Sentences

The most significant challenge was dealing with neutral sentiment in reviews. The datasets lacked sufficient labels for neutral sentences, making it difficult for the model to generalize. The key to addressing this issue was to retain as many words as possible rather than removing them with the help of pre-defined lists such as stop words, thereby maintaining context of these neutral words.

Other Challenges

Additional challenges included managing hardware and computational limitations, which significantly slowed down the project.

5. Conclusion

While some experiments led to improvements, others added unnecessary complexity leading to slower training time. The final model architecture, consisting of the Nadam optimizer, Bidirectional LSTMs, and the strategical preprocessing, balanced the processing and performance.

The findings of the deep learning-based approach for sentiment analysis indicate that the Bidirectional LSTM architecture outperforms the LSTM architecture, as it allows the model to handle contexts. Additionally, Expanding the vocabulary improved contextual understanding, requiring adjustments to the stopwords list.

Furthermore, named entity removal proved beneficial, as many entities do not contribute to sentiment in terms of context. However, since this process is computationally expensive, shifting it to the tokenization phase helped reduce computational overhead.

Additionally, employing neutral sentiment can enhance the evaluation, as many words carry neutral sentiment unless they are used in a specific context. Due to the binary nature of the training dataset, the model assigns sentiment to these neutral words during its training, which affects the model's predictions.

In conclusion, this project has provided valuable insights into the process of sentiment analysis and ways to enhance it.

The project code, datasets, and additional files are available at the following GitHub repository: https://github.com/NasiriMohsen/Keras_Movie_Sentiment_Analysis

References

Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A., & Potts, C. (2013). Recursive deep models for semantic compositionality over a sentiment treebank. Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, 1631–1642. Association for Computational Linguistics. https://www.aclweb.org/anthology/D13-1170

Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., & Potts, C. (2011). Learning word vectors for sentiment analysis. Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, 142–150. Association for Computational Linguistics. http://www.aclweb.org/anthology/P11-1015

Pang, B., & Lee, L. (2005). Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. Proceedings of the Association for Computational Linguistics (ACL). https://huggingface.co/datasets/cornell-movie-review-data/rotten_tomatoes

Arize AI. (n.d.). Movie reviews with context drift [Dataset]. Hugging Face. Retrieved January 29, 2025, from https://huggingface.co/datasets/arize-ai/movie_reviews_with_context_drift

itsabba3. (n.d.). IMDB rating dataset [Dataset]. Kaggle. Retrieved January 29, 2025, from https://www.kaggle.com/datasets/itsabba3/imdb-rating