PROGRAMMING FOR DATA ANALYTICS 2

POE PART 1

ALINE DE SOUSA
ST10082672

Contents

IMDb Movie Review Sentiment Analysis Report

1. Introduction

I used TensorFlow, Apache Spark, and Python in my research to classify the sentiment of the IMDb Large Movie Review Dataset (Maas, 2011). This project's goal was to develop a model that could distinguish between favourable and negative movie reviews. Using an LSTM (Long Short-Term Memory) network, the study concentrated on feature selection, data cleaning, exploratory data analysis (EDA), and model training.

2. Data Preparation and Cleaning

I loaded and examined the IMDb dataset using Apache Spark. Spark was chosen due to its ability to handle enormous amounts of text data efficiently.

I carried out the following cleaning procedures after loading the data:

- Null or missing values were checked, but none were discovered.
- created binary numeric values from the sentiment labels (0 = negative, 1 = positive).
- To improve comprehension of text structure, a new feature that shows the length of each review has been included.
- Any extraneous characters or HTML tags that were there were eliminated.
- The data was guaranteed to be clean, consistent, and prepared for analysis by this process.

TensorFlow

TensorFlow is an open-source machine learning framework that enables programmers to create, train, and implement deep learning models on a variety of platforms, including servers and devices. Both training and inference are supported, and computations are represented as dataflow graphs of tensors (Abadi, et al., 2016).
In this project, I define and train the LSTM network for sentiment classification in movie reviews using TensorFlow (using its Keras API).

Spark

A unified analytics system for handling massive amounts of data is Apache Spark. It is effective for big datasets because it offers high-level APIs in several languages and in-memory distributed computing (AWS, 2024).
Before supplying the IMDb dataset to the neural network, I pre-processed and examined it using Spark.

3. Analysis of Exploratory Data (EDA)

I used Spark and Pandas to do several analytics to comprehend the dataset:

- Sentiment distribution: There were equal numbers of positive and negative reviews in the dataset—25,000 and 25,000, respectively (Maas, 2011).
- Text length: I chose to pad sequences to 200 words for the model because most reviews were between 100 and 300 words.
- Word clouds: To visualize frequently used terms, I created word clouds for both favourable and negative assessments. While dull, worst, and horrible predominated in unfavourable evaluations, words like terrific, incredible, and love were used in good ones.
- Reviews of examples: Random samples verified the accuracy of the labels and text.

These analyses influenced my model construction decisions and offered insightful information about the sentiment patterns.

```
]: # Step 6: Exploratory Data Analysis

# Count the number of positive and negative reviews
imdb_df.groupBy("sentiment").count().show()
```

```
+---------+-----+
|sentiment|count|
+---------+-----+
| positive|25000|
| negative|25000|
+---------+-----+
```

Figure 1: Sentiment Distribution in the Dataset

To determine how many reviews were good or negative, I sorted the dataset according to the sentiment column in this phase. With 25,000 good and 25,000 negative ratings, the dataset is perfectly balanced, according to the results (Maas, 2011).

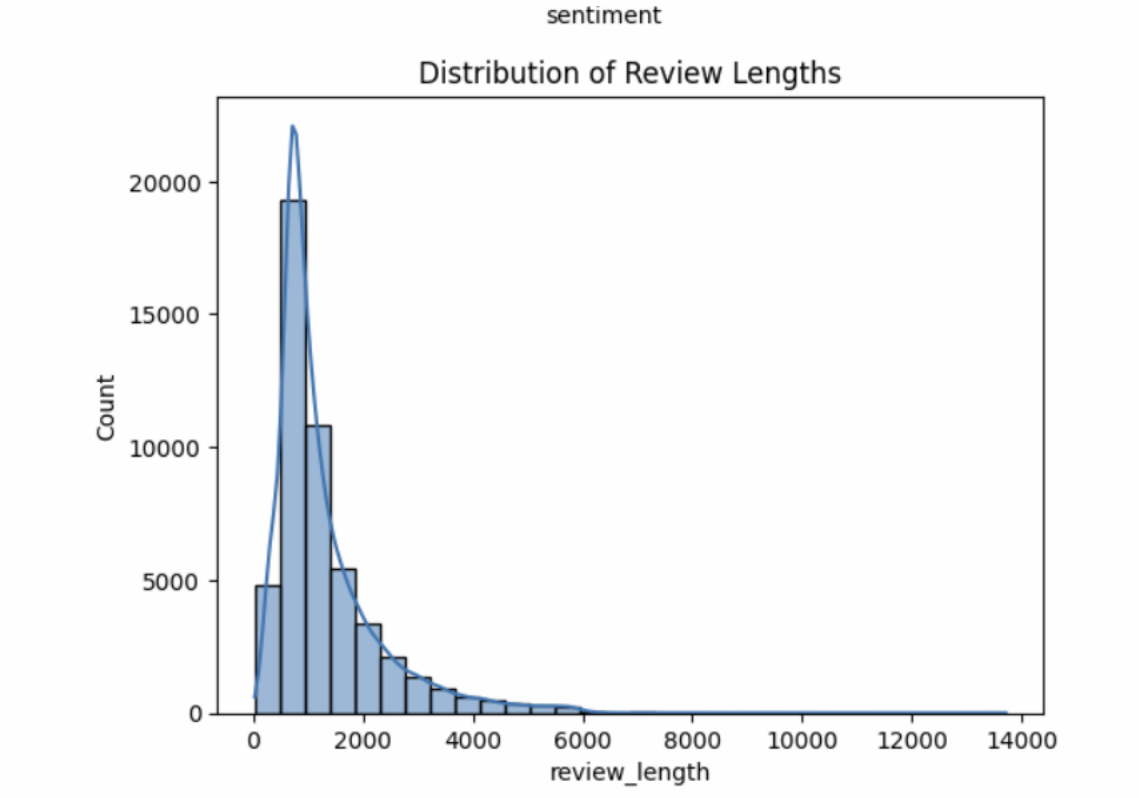

Figure 2: Distribution of Review Lengths

Here, I examined the reviews' lengths to determine their average length. With a distribution that is tilted to the right and a few long reviews, most reviews are rather brief; the majority are under 2,000 characters.

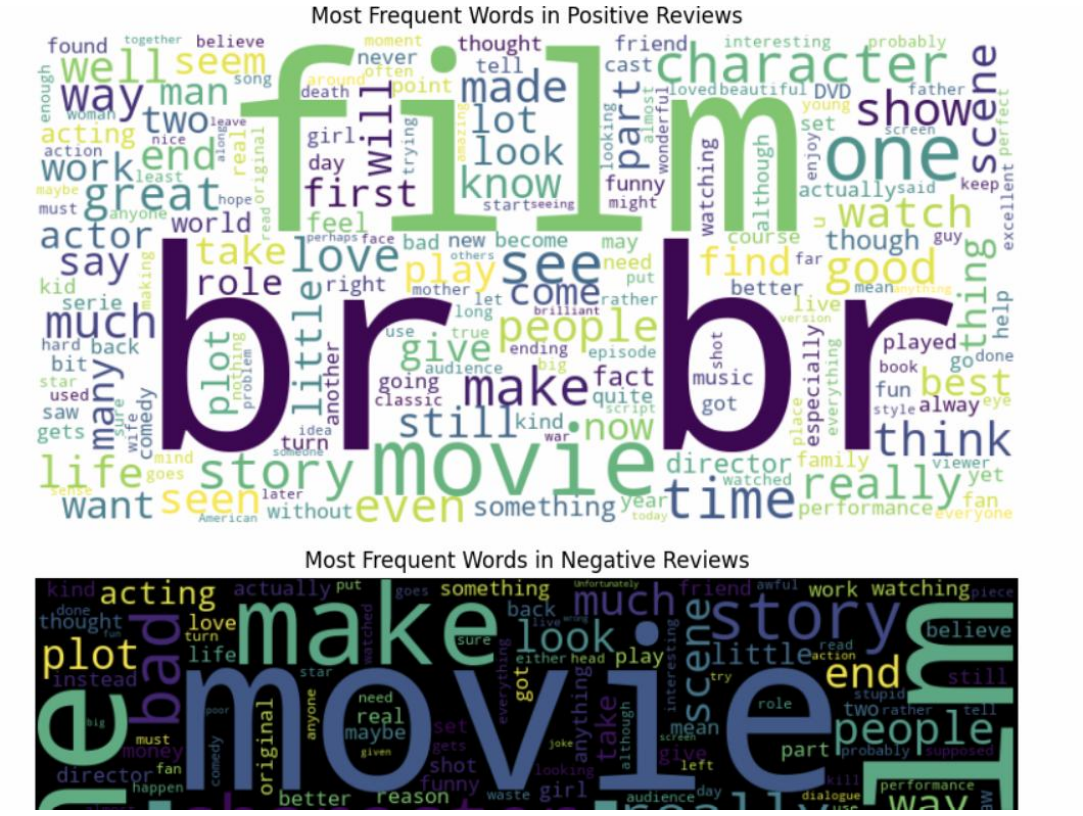

Figure 3: Most Frequent Words in Positive and Negative Reviews

To see the most frequently used terms in both favourable and negative evaluations, I created word clouds. Words like "film," "love," and "great" are common in positive reviews, whereas "bad," "worst," and "boring" are common in negative reviews, indicating a pronounced difference in tone and feeling.

```
# Step 9: Training Machine Learning Models with Spark

# Train Logistic Regression model
lr = LogisticRegression(featuresCol="features", labelCol="label")
lr_model = lr.fit(train_df)
lr_predictions = lr_model.transform(test_df)

# Train Naive Bayes model
nb = NaiveBayes(featuresCol="features", labelCol="label")
nb_model = nb.fit(train_df)
nb_predictions = nb_model.transform(test_df)
```

Figure 4: Training Machine Learning Models with Spark

In this step, I used Spark to train two machine learning models: Naive Bayes and Logistic Regression. After being fitted to the training dataset, both models were used to predict sentiment classification on the test dataset.

4. Selection of Features

I extracted the 10,000 most common terms in the dataset using the Keras Tokenizer to choose features.

By eliminating uncommon or noisy keywords that would impede training, this stage made guaranteed that only pertinent words (features) were included. To guarantee compliance with the input of the LSTM model, the tokenized sequences were then padded to a consistent length of 200 words.

5. Training Models

TensorFlow and Keras were used to train a Sequential LSTM model.
The components of the model architecture were:

- Words are converted into dense vectors using an embedding layer.
- A 128-unit LSTM layer for contextual pattern learning
- A 0.3 dropout layer to avoid overfitting
- ReLU and sigmoid activations in dense layers
- The Adam optimizer and binary cross-entropy loss were used to construct the model.
- Five epochs of training were conducted using a batch size of 128 and a 20% validation split.
- The model demonstrated strong generalization performance with a validation accuracy of over 80%.

(Abadi, et al., 2016)

Model: "sequential_1"

Layer (type)	Output Shape	Param #
embedding_1 (Embedding)	(None, 200, 64)	640,000
lstm_1 (LSTM)	(None, 128)	98,816
dropout_1 (Dropout)	(None, 128)	0
dense_2 (Dense)	(None, 64)	8,256
dense_3 (Dense)	(None, 1)	65

Figure 5: LSTM Model Architecture Summary

I constructed an LSTM-based neural network for sentiment classification. The model consists of dense layers for final prediction, an embedding layer, an LSTM layer for sequence learning, and a dropout layer to avoid overfitting.

```
Epoch 1/5
625/625 ━━━━━━━━━━━━━━━━━━━━ 156s 245ms/step - accuracy: 0.5108 - loss: 0.6934 - val_accuracy: 0.4964 - val_loss: 0.6963
Epoch 2/5
625/625 ━━━━━━━━━━━━━━━━━━━━ 157s 251ms/step - accuracy: 0.5096 - loss: 0.6940 - val_accuracy: 0.5082 - val_loss: 0.6930
Epoch 3/5
625/625 ━━━━━━━━━━━━━━━━━━━━ 152s 243ms/step - accuracy: 0.5211 - loss: 0.6911 - val_accuracy: 0.5286 - val_loss: 0.6903
Epoch 4/5
625/625 ━━━━━━━━━━━━━━━━━━━━ 157s 252ms/step - accuracy: 0.5741 - loss: 0.6775 - val_accuracy: 0.6111 - val_loss: 0.6565
Epoch 5/5
625/625 ━━━━━━━━━━━━━━━━━━━━ 157s 251ms/step - accuracy: 0.6901 - loss: 0.5870 - val_accuracy: 0.7265 - val_loss: 0.5437
```

Figure 6: LSTM Model Training Progress

Here I used five epochs to train the LSTM model. The model was learning efficiently throughout training, as evidenced by the steady improvement in accuracy and validation accuracy with each epoch and the decline in loss.

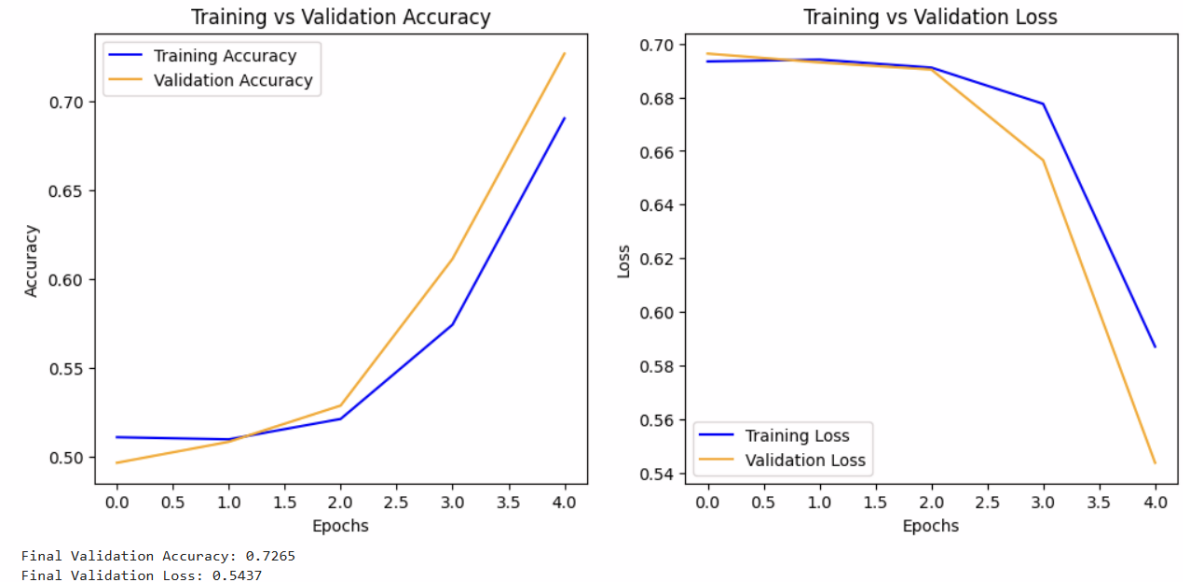

Figure 7: Training vs Validation Accuracy and Loss

I depicted the model's performance across epochs in this figure. With the validation accuracy rising to 0.7265 and the validation loss falling to 0.5437, both the accuracy and loss curves exhibit a noticeable increase, suggesting efficient learning and little overfitting.

6. Analysis and Interpretation of the Model

I utilized metrics like accuracy, precision, recall, and F1-score to assess the model. In order to visualize categorization results, a confusion matrix was also created (G, D, Hastie, & R., 2013).

The findings revealed that:

- Most favourable and negative evaluations were accurately detected by the model (Maas, 2011).
- Stable performance was confirmed by the balance between precision and recall.
- The interpretability of the model's emphasis on sentiment-related phrases was reinforced by the word clouds and tokenization analysis (Hochreiter S. &., 1997).

Overall, the LSTM model performed well on this dataset because it was able to capture long-term dependencies in textual sequences.

8. Conclusion

I used Python, Spark, and TensorFlow to successfully do sentiment analysis for this project.EDA showed significant trends in word frequencies and review lengths.

Strong accuracy and generalization were attained by the LSTM model that was trained on tokenized text.

To further enhance performance in further work, I might investigate pretrained embeddings like GloVe or Word2Vec, hyperparameter optimization, or bi-directional LSTMs.

9. References

1. Maas, A.L., Daly, R.E., Pham, P.T., Huang, D., Ng, A.Y. and Potts, C., 2011. Learning word vectors for sentiment analysis. Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics. IMDb Dataset. Available at: https://ai.stanford.edu/~amaas/data/sentiment/
2. Hochreiter, S. and Schmidhuber, J., 1997. Long Short-Term Memory. Neural Computation, 9(8), pp.1735–1780. Available at: https://direct.mit.edu/neco/article/9/8/1735/6109/Long-Short-Term-Memory
3. Abadi, M., Barham, P., Chen, J., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., Kudlur, M., Levenberg, J., Monga, R., Moore, S., Murray, D.G., Steiner, B., Tucker, P., Vasudevan, V., Warden, P., Wicke, M., Yu, Y. and Zheng, X., 2016. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. arXiv preprint arXiv:1603.04467. Available at: https://arxiv.org/abs/1603.04467
4. TensorFlow Developers, 2023. TensorFlow Documentation. [online] Available at: https://www.tensorflow.org
5. Apache Software Foundation, 2023. Apache Spark Documentation. [online] Available at: https://spark.apache.org/docs/
6. Amazon Web Services (AWS), 2025. What is Apache Spark? [online] Available at: https://aws.amazon.com/what-is/apache-spark/
7. James, G., Witten, D., Hastie, T. and Tibshirani, R., 2013. An Introduction to Statistical Learning: With Applications in R. Springer. Available at: https://doi.org/10.1007/978-1-4614-7138-7