

A Comparative Analysis of Naïve Bayes and LSTM for Sentiment Classification in Text Data

Author: Arman Afshari-Rahimzadeh

Contents

I. Introduction.....	3
1. Introduction to the Domain-Specific Area.....	3
2. Objectives of the Project	3
3. Description of the Selected Dataset	4
4. Evaluation Methodology	5
II. Implementation	6
5. Data Preparation	6
6. Baseline Performance.....	9
7. Comparative Classification Approach	11
III. Conclusions.....	16
8. Performance Analysis & Comparative Discussion.....	16
9. Project Summary and Reflections.....	16
References	18
Appendix	19

Introduction

1.1 Introduction to the Domain-Specific Area

In the most basic sense, text classification can be described as a function that takes a text input and gives a set of pre-established labels as output. It can be used for a variety of tasks such as spam detection, sentiment analysis, topic classification and more. The past few years have seen a boom in the field of Natural Language Processing (NLP) and resulting in the creation of a multitude of models geared towards the interpretation of text based data.

Statistical models, including Naive Bayes and Logistic Regression, have always enjoyed some popularity owing to their efficient and uncomplicated nature. They depend on feature extraction methods such as Bag-of-Words (BoW) and Term Frequency-Inverse Document Frequency (TF-IDF), both of which aid in translating the text into a numerical format. While these models are indeed effective to a certain extent, they perform poorly when it comes to understanding the intricate bonds between the words in a given text (Levity, 2024)

The development and creation of deep learning models like Long Short-Term Memory (LSTM) networks have transformed the realm of NLP completely. To put it simply, LSTMs use word embeddings to factor in the context while deep learning architectures extract meaning from the words, both of which significantly improve the performance on more complicated models. It is necessary to analyze and compare these two methodologies to gain clarity regarding their respective strong points, weaknesses, as well as real-world applications (Dataiku, 2024)

Comparative analysis will be performed with LSTM and Naive Bayes about imbued datasets. So I will be classifying the sentiment of the reviews into positive or negative.

1.2 Objectives of the Project

The overarching goal of this research work is to, first, implement the intended models (Naive Bayes and LSTM) and evaluate the performance differences between the two. This research work seeks to understand the limitations, effectiveness, and applicability of the text models in the domain of text classification, with a primary focus on sentiment analysis tasks. The purpose of comparing the models is the purpose of seeking insight into which model would best serve the purpose of text classification.

The model is simpler to implement, highly computationally efficient, and achieves quite satisfactory results, thus fueling its case to be employed for building a benchmark comparison. This is due to the fact that Naive Bayes works seamlessly with word counts. The model performs the classification of texts by relying on the representation of features through Term Frequency-Inverse Document Frequency (TF-IDF), further putting weight over the words present in the document depending on the entire database. On the other hand, In comparison to the LSTM model - that generates embeddings using supervised learning which focuses on sentences rather than the individual words due to the use of recurrent neural networks - this model captures contextual and semantic information in the text (Nikmah, 2022).

The project's primary goal is to evaluate the models' advantages and disadvantages. As a result, the models are assessed for their ability to meet text syntactic and semantic requirements using metrics such as accuracy, precision, recall, and F1-score. In addition, the research provides insights on the

effectiveness of these models in real life, for example, analyzing customer feedback, keeping track of sentiments, or recognizing spam.

The goal of the model is also intended to help those who seek to put the task into practice, especially justifying the type of model they can rely on according to the capacity of the task, the data set, and the computing size. This way, it helps to balance the modeling trade-off knowledge of the shallow approach and the deep learning approach along the lines of context modeling, pushing NLP further.

To sum up, the project seeks to fulfill the research gap by identifying the contribution of Naive bayes and LSTM models in text classification. The results are useful to researchers and practitioners as they assess the rewards and flaws of these approaches and inform better decisions about NLP tasks (ActiveWizards, 2024).

1.3 Description of the Selected Dataset

This distinct dataset is crucial for the analysis because it facilitates a fair contrast between the embedding and statistical models. In this instance, the Nielsen Dataset of reviews was selected as it can serve as a standard for sentiment classification tasks.

The dataset contains 50,000 reviews that are segregated into two classes making it easier for a supervised classification model: whether the review is negative or positive. These classes are balanced, which means that they bolster in equal proportions thereby eliminating the bias for either positive or negative reviews. This balance ensures that the assessment of the compiled models is effective. There are a wide range of complexity, writing styles and even lengths associated with these reviews which makes room for the models to grow stronger during the analysis (Lakshmi, 2018).

Only two parameters are associated with this dataset as the structure is rather simple, review along with sentiment. With the review being cluttered, HTML tags, special characters and even improper capitalization add to the noise which is already apparent in the data. These are labeled as categorical data while sentiment is represented through textual labels. So, the proper analysis would require an appropriate formatting of the data which is only possible through proper preparation.

This dataset was sourced from Kaggle as part of a sentiment analysis benchmarking project and is publicly accessible. This is evidence that the data is relevant, real-world and diverse. It is a suitable dataset for evaluating traditional statistical models and deep learning models because it has a good mix of class populations and it is easier to access.

The attributes within the dataset also pose some problems which need to be addressed by the models. For example, the length of reviews is not constant, which needs trimming or padding for deep learning models, or there is noise in the data, which requires other advanced cleaning methods for statistical and embedding based models. All these are reasons that the IMDb dataset is suitable for use when assessing the performance of Naive Bayes against LSTM models.

In conclusion, the IMDb dataset is adequate for this project. The characteristics of the data might be highly unbalanced, however, it is highly applicable, and organized which can lead to highly admissible results. The characteristics of the dataset also add variety to the analysis done on the two model types, which further proves that it is appropriate for the research.

1.4 Evaluation Methodology

When comparing the outcome of the Naive Bayes and LSTM models, certain key performance indicators were employed to assist in achieving the goal of this study in the most accurate way. These metrics provide a comprehensive understanding of the models' strengths and weaknesses in text classification tasks. They are also combined with more complex measures such as ROC-AUC for better understanding of the models' classification performance to derive deeper insights into most of the models' classifications capabilities.

They are also combined with more complex measures such as ROC-AUC for better classifiers insight. Accuracy is usually regarded as a basic indicator of the extent of correct classifications that were made from the total samples taken. However, accuracy alone can be misleading particularly for skewed datasets as it does not reflect the number of positive and negative classes that are present. In this project, where the dataset is balanced, it is clear that accuracy is a reliable indicator of performance.

Furthermore, these include Precision and Recall, which are more complicated indicators, that will also be discussed below. Precision is defined as the ratio between true positives and predicted positives and it investigates how well the model classifies positive samples among all tested. This metric is important in situations when there is a high penalty for inaccurately judging something as a positive, such as, spam detection. On the other hand, recall provides an indication of the extent to which the model detects all the existing actual positive samples and is expressed as the ratio of true positives to actual all positive: $\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$. Recall is highly important in situations where some positive samples should be retained without failure, for instance medical diagnosis.

F1 score works on the principle of incorporating both recall and precision for model gradation and thus can be seen as the average between the two. It is especially useful when we need to strike a balance between that, recalling something particular F1 score tends to make that particularly egregious more so when looking at this model performing equally well across both positive and negative sides. In this project, F1-score assists in achieving a broader and more comprehensive comprehension of how the models behave across both sides of the class and not just positive or negative.

Secondly to complement and assist the models in validating the distinguishing of a negative to a positive review, ROC (Receiver Operating Characteristic) - AUC (Area Under Curve) is looked into as it helps with the validation. Essentially, the concept behind the use of ROC and AUC is two fold: first, it is heard that Kurakin combines true positive rate (recall) and false positive rate into a single model presented as a graph, called ROC curve, and in its true sense, AUC does provide quite the edge with separated classes as a model comparison tool.

In a more diverse range, skeptics are known to apply all of these metrics, especially the AMS and LSTM models, on a wider range, as previously mentioned. Thanks to the balanced dataset, there is little in the way of distinguishing between F1, recall, precision as their weightage while working is uniform in nature providing an all rounded approached. These visuals alongside confusion matrices, precision-recall curves, and ROIs help assess the two methods against one another, ultimately helping determine each methods pros and cons.

In conclusion, the chosen evaluation methodology provides a detailed assessment of the models' performance. By focusing on multiple metrics, this approach ensures a nuanced understanding of how each model handles text classification, making the comparison both robust and meaningful.

The word cloud for positive reviews contains the words 'movie', 'good', 'love' and 'great' which is also the common words used when writing a review with a positive sentiment.

<p>The corrected textual data was presented in terms of numerical feature vectors through Term Frequency-Inverse Document Frequency (TF-IDF). Documents containing words are weighted by their frequencies – TFIDF’s -importance. In order to enhance the speed of computation, the size of the vocabulary was constrained to the most frequently occurred 5000 words.</p>	<p>The prepared text was converted to a sequence of integers by assigning each word in role of character a number in the vocabulary. These sequences were cut or expanded to target length of 200 words to guarantee that all the model’s inputs were of the same such length. The semantic relationships of words via training were established whereby word embeddings were dedicated.</p>
--	--

Handling Review Lengths:

In the case of the dataset extracted from IMDb, I would like to note that the review lengths differ to a great extent with a number of reviews being concise while others are lengthy. Such existence of variation poses an impediment to a statistical and deep learning model and thus calls for special preprocessing methods.

As a statistical model, Naive Bayes can consider the entire review as a feature vector as long as there is a word frequency representation in the form of T-F IDF. On the other hand, deep learning models such as the LSTM accept inputs of a specific arbitrary length. Inevitably, this calls for a solution. To this end, shorter reviews were zero padded while the longer ones were truncated to a maximum length of 200 words. This strategy guarantees uniformity without losing vital details for sentiment analysis.

The lengthening trend of the reviews is graphed below. It bears out how often shorter reviews are likely to appear and the long hug horizontal graphs of very long reviews. This trend influenced the decision to maintain a fixed sequence length in order to promote efficiency.

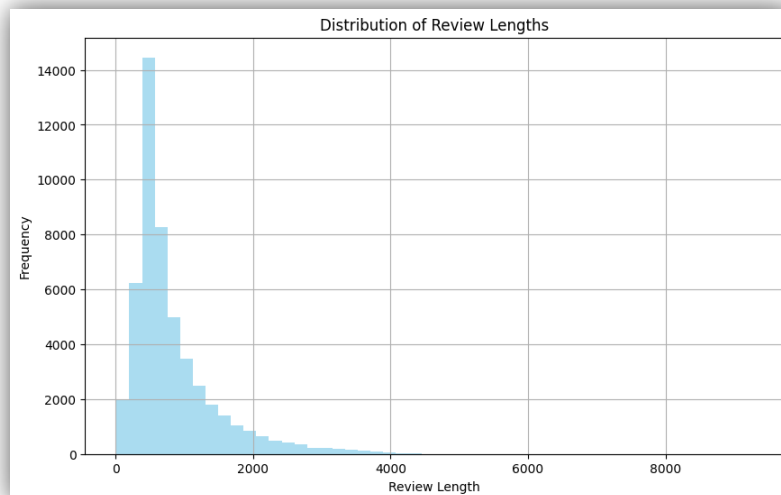


Figure 3: Distribution of Review Lengths

Changing the length of the reviews was important to the pre processing pipeline because it enabled the Naive Bayes and the LSTM models to operate on the same text while still taking advantage of their different capabilities for sentiment classification tasks.

Rationale for Preprocessing Steps

All the preprocessing steps were intended to solve specific dataset problems in a more optimal way:

- Removing noise and stopwords increases the power of features
- Tokenization converts text into small and relevant units for further evaluation.
- Transformations such as TF-IDF and embeddings set the stage for various modeling techniques where statistical models are driven by features and deep learning models utilize context and meaning.

The preprocessing pipeline was constructed in terms of the needs posed by the Naive Bayes and LSTM models. Reasonable steps were taken to ensure clean, coherent, and properly formatted data that significantly enhanced the quality of the models as well as the evaluation of the models. These steps showed how differently the two models represented the text.

2.2 Baseline Performance

The Naive Bayes model with TF-IDF vectorization of features has been taken as a baseline of this project. This baseline was used because general Naive Bayes algorithms are a common tool for the classification of texts in most text classification tasks including sentiment analysis. It is also a popular starting point in most studies because it is simple, quick and gives reasonable results for high dimensional spaces. Adoption of this baseline has fulfilled a logical objective allowing for more advanced LSTM models to have baseline measures that made sense and were interpretable.

Ease and Speed of Execution:

Naive bayes is easy to code, requiring little IT effort making it suitable for prototyping purposes. It also works efficiently on other numericalized text data such as in conjunction with TF-IDF feature extraction models.

Considered as a Baseline Model:

In the area of NLP, Naive Bayes is frequently recommended as a baseline for cross – comparison for other models since it is a core model in text categorization.

Fit to the Dataset:

The IMDb giving two classes only, is well suited for the task since it is structured in a way to comfortably fit the naive bayes approach. Due to the basic nature of the structure of the dataset (review texts combined with sentiment labels), it is perfect for the algorithm to work without doing excessive feature engineering.

Relevance for Comparison:

Since Naive Bayes is a popular classical statistical method it is possible to show side by side the disadvantages and advantages of the learning embedding-based LSTM model. Such a comparison elucidates the strengths and weaknesses of statistical approaches versus deep learning ones when dealing with text.

It was implemented in the following way:

- To reduce processing time, TF-IDF vectorization was used to create 5000 unique words in 5000 word vocabulary for the text data.
- A Multinomial Naive Bayes classifier was trained on 40,000 reviews (80% of the dataset) and tested on 10,000 reviews (20% of the dataset).
- The model was assessed using primary evaluation parameters such as accuracy, precision, recall, and F1-score.

Baseline Results

The Naive Bayes model was able to achieve an accuracy of 85%, which is a fairly good performance for a classical statistical technique. A similar 85% tally was attained for the other key parameters such as precision, recall and F1-scores, as pertains to positive and negative sentiments. These results underscore the capability of the model when it comes to working with high-dimensional vectorized representations of text data using TF-IDF for instances.

Getting an index of all the tweets, over a large enough set context does have issues, such as the sentiment. While relying solely on a model, meant to classify a sentiment, treating the words individually isn't a clear context sentiment.

A comprehensive perspective of the model's output can be acquired from the analysis of the confusion matrix presented below:

- **True Negatives:** 4,311 reviews correctly classified as negative.
- **True Positives:** 4,465 reviews correctly classified as positive.
- **False Positives:** 650 reviews incorrectly classified as positive.
- **False Negatives:** 574 reviews incorrectly classified as negative.

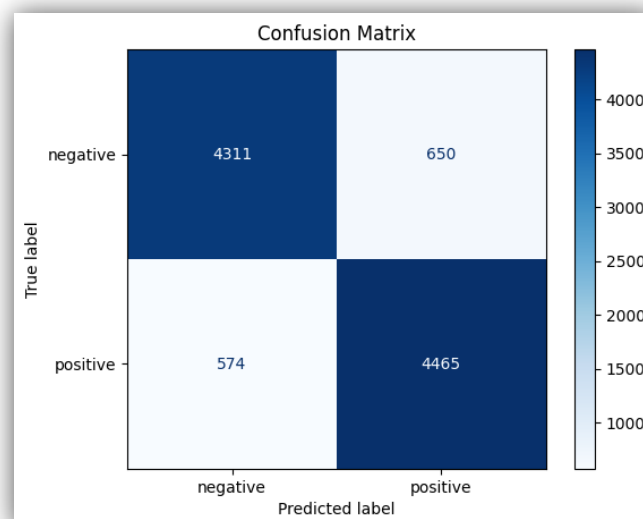


Figure 4: Confusion Matrix – Visualization of model performance

The confusion matrix also supports the argument that the models assisted by evolving algorithms outclass Naive in sentiment predicting accuracy, nevertheless, like the NNM, the Naive does not take nuances in word pairing into account when working with complex text data.

On the other hand, the simple naive takes into consideration upto two phrases of words. Therefore, the model provides a decent anchor, demonstrating the benchmark for the the other evolving LSTM model to jump over, as a model designed for multilayer networks. As And when it comes multi layered networks containment area, the anger does put out closer to the context alongside it's shortcomings.

Meaningful Benchmark for Comparison

The performance of the Naive Bayes model establishes such benchmarks for text prediction tasks since it utilizes primary techniques but does not use any complex methods. This explains the performance of LSTM model which utilizes word vectorization and sequence processing to model context.

To sum up, the Naive Bayes model is assessed as a good standard, allowing us to evaluate the LSTM model more comprehensively in terms of its merits or its contribution. This standard guarantees that the effectiveness achieved can be only ascribed to the sophisticated characteristics of the embedding-based technique with no reference to differences in preprocessing and evaluation.

2.3 Comparative Classification Approach

In this paper, we investigate two methods of text classification, a Naive Bayes model, a conventional statistical approach, as well as a Long Short-Term Memory LSTM which is a neural network based deep learning model. Text classification of movies from the Internet Movie Database IMDb was undertaken using both approaches, paying particular attention to their structural designs.

	Naive Bayes (Statistical Model)	(LSTM) - Modern Deep Learning Model
Architecture	Naive Bayes uses the Bayes' theorem to derive the probability of a class based on the observed features. It makes the strong assumption that words are independent, that is, the presence of one word is not dependent on the presence of another. TF-IDF (Term Frequency-Inverse Document Frequency) was implemented to convert text information into numeric features.	<p>The Long Short-Term Memory (LSTM) model, an advanced variant of Recurrent Neural Networks (RNN), overcomes the vanishing gradient problem and is highly effective for sequential data. Its ability to capture long-term dependencies and contextual relationships in text makes it ideal for sentiment classification.</p> <p>The architecture utilized in this research includes an embedding layer that transforms tokens into dense vectors of size 128, enhancing word relationships. Two LSTM layers, with 128 and 64 units respectively, are stacked sequentially to process the data</p>

		effectively. Dropout layers are incorporated after each LSTM layer to prevent overfitting. Finally, a dense layer with a sigmoid activation function outputs the probability of positive or negative sentiment, enabling accurate classification.
Implementation	<p>Preprocessing: The text was preprocessed through tokenization, cleaning and then vectorization based on TF-IDF. This time vocabulary was also used but it was capped at 5000 words.</p> <p>Model: the Multinomial Naive Bayes classifier from the sklearn library was applied as it is known to be appropriate for text data represented by the counts of words or their frequencies.</p>	<p>Preprocessing: Text reviews were tokenized and converted into sequences of integers. Each sequence was padded or truncated to a fixed length of 200 words for uniform input.</p> <p>Model: Implemented using the TensorFlow library with the Sequential API.</p>
Training	<p>The Cutting Edge model was trained for 40,000 reviews (which is 80% of the total reviews).</p> <p>Since it is a simple model, it was quite cost efficient in its training and was completed in seconds.</p>	<ul style="list-style-type: none"> • Dataset: The model was trained on 40,000 reviews, with 20% of the training data reserved for validation. • Loss Function: Binary cross-entropy. • Optimizer: Adam optimizer. • Epochs: 5, with a batch size of 32.
Optimization	Optimization was lower than we assumed in advance. Alpha, the smoothing parameter in the prior was maintained at its default value to cater for previously unseen words.	<ul style="list-style-type: none"> • Regularization techniques such as dropout were applied to mitigate overfitting. • Validation loss was monitored during training, and early stopping could be employed for further optimization in future iterations.
Strengths	<ul style="list-style-type: none"> • Profiting and memorizing both were quite time efficient. • Complexity reduced to achieve better performance while increasing accuracy. • TF-IDF representations worked quite well. 	<ul style="list-style-type: none"> • Captures context and semantics through word embeddings. • Effectively handles sequential dependencies within the text. • Scales well to larger datasets and more complex language structures.
Weaknesses	<ul style="list-style-type: none"> • Word relationships cannot be captured because word independence is taken as an assumption. 	<ul style="list-style-type: none"> • Computationally intensive, requiring more time and resources for training. • Prone to overfitting without proper regularization techniques.

	<ul style="list-style-type: none"> The meaning and context of the text is difficult for this model to understand. 	
--	--	--

The Training and Validation Accuracy and Loss Curves illustrate the process of acquiring knowledge of the particular model:

Training and Validation Accuracy:

As seen in the graph, accuracy for the training set and the validation set improved consistently with increase in epochs, indicating that the model is able to perform generalization.

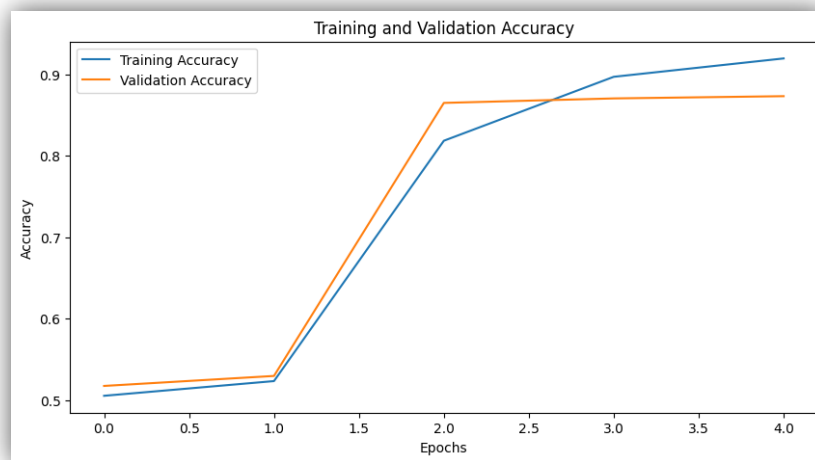


Figure 5: Training and Validation Accuracy

Training and Validation Loss:

From the above graphical representation of loss, it can be observed that both the training and testing loss had a gradual decline, this means there was learning. There are signs of slight overfitting towards the end in further epochs.



Figure 6: Training and Validation Loss

Comparison of Performance

The evaluation of the two models, Naive Bayes and LSTM, in terms of their abilities to perform sentiment classification shows a marked difference in performance. Several measures including accuracy, precision, recall, F1-score, and ROC-AUC were used to measure their performance, which is shown in the final table:

Metric	Naive Bayes	LSTM
Accuracy	85%	88.13%
Precision	85%	88%
Recall	85%	88%
F1-Score	85%	88%
ROC-AUC	Lower (not measured)	0.95
Training Time	Seconds	Several minutes
Context Handling	Ignores context	Captures context
Scalability	Suitable for small tasks	Suitable for complex tasks

The Precision-Recall Curve presented below clearly indicates that the LSTM model has a better performance than other predictive models because it integrates precision and recall at the different thresholds. Apart from this, the curve elucidates that LSTM has a higher precision value while in various recall levels whenever it is juxtaposed to Naive Bayes, indicative of its capabilities in resolving conflicting and intricate cases in sentiment analysis classification.

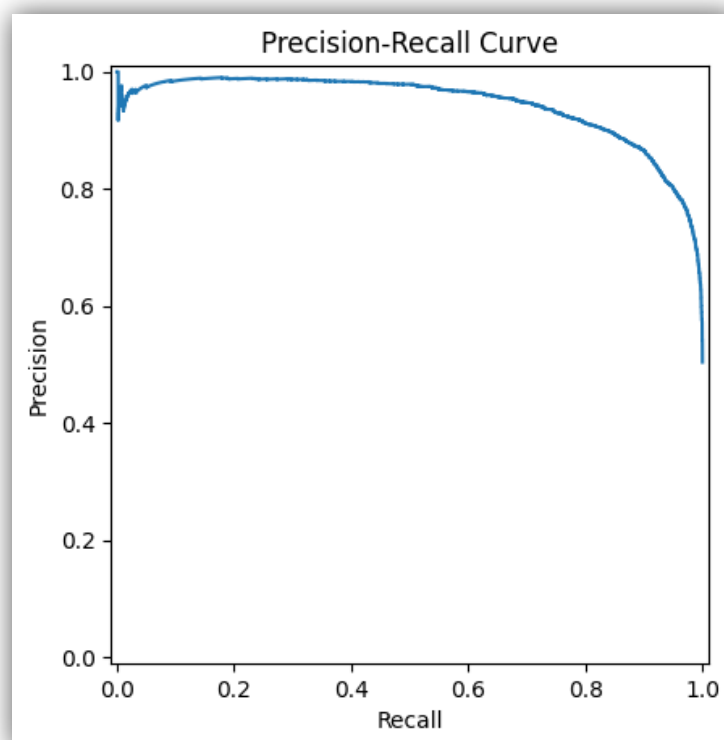


Figure 7: Precision-Recall Curve

The ROC Curve presented below also supports the evidence of LSTM exceeding other models in class separation ability, as far as this one has an AUC of 0.95. This metric is illustrative on the extent to which the model can be able to tell apart two contrasting models which are positive and negative emotions, further showcasing a semblance of the model's efficacy in tasks that only has two classifications.

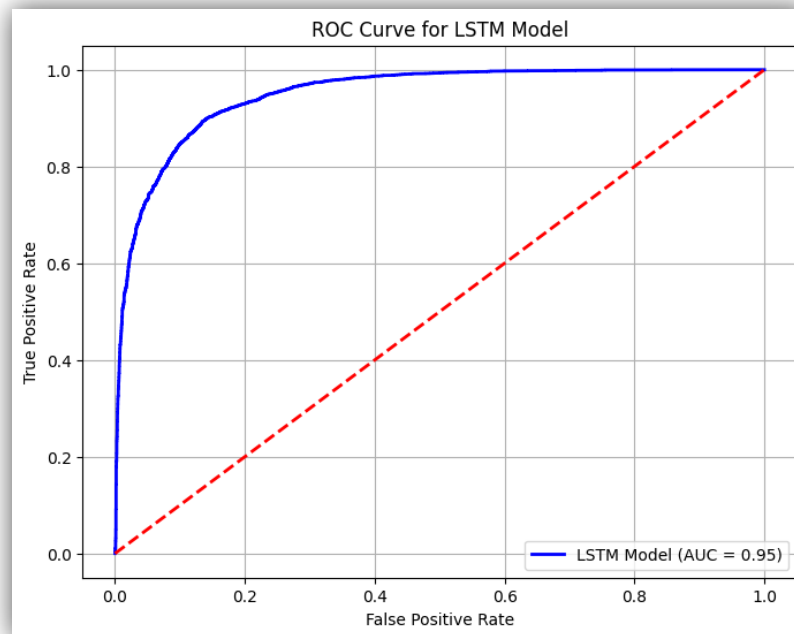


Figure 8: ROC Curve for LSTM Model

Lastly, the accuracy of the sentiment classification model is further enhanced with the combination of the precision recall and ROC curves as it is able to trump naïve bayes especially in use cases involving intricate and all encompassing data. LSTM on the other hand uses significantly more resources and as a result is more temporal in nature than Naïve bayes LSTM is able to retain more information sequentially making it ideal for classifying large scale and intricate sentiment classification tasks.

Conclusions

3.1 Performance Analysis & Comparative Discussion

The juxtaposition that exists between the Naive Bayes and LSTM enables us to see their characteristics better. The Naive Bayes model performed consistently, achieving an 85% accuracy as a statistical approach. The positive and negative classes also had an F score of 85%. The model was also computationally cheap and easy to use which allowed it to be easily used for tasks with a low quantity of resources. The TF-IDF representation enhanced its robustness for high-dimensional text data, however the major drawbacks of the model were its understanding of the context as well as the word independence and limited necessity. Even though it performed well for lower quantities of resources, its performance started deteriorating with larger datasets.

Moving on to the LSTM model, this achieves accuracy of 88.13% with an improved F score of 88 across the ballpark metrics showing a clear sign to outperform Naive bayes. It also showed a clear cut improvement against LSTM's ROC-AUC score of 0.95. LSTM captures the contextual interdependencies between embeddings and time series hence it easily scales to more sophisticated language applications along with more data. But there is a caveat, these benefits have implications of far more expensive training duration and resources. The level of regularization, dropout layers in this case, has a significant impact on the learning performance of the LSTM approach and if proper techniques are not employed the LSTM model ends up becoming more robust to overfitting.

This work sheds light on the compromises between complexity and contextual awareness. Naive Bayes is an adequate and fast mechanism to put into action as a benchmark for easier applications but the application of LSTM is much more promising for handling complex text classification problems. Each of the applications has its purposes in line with the director, type of task and the computer system.

Key Takeaways

- The Naive Bayes model is a robust choice for simpler tasks with limited computational resources. Its speed and interpretability make it valuable for quick sentiment classification.
- The LSTM model is better suited for complex datasets and tasks where capturing context and word relationships is essential. Its superior performance, as seen in this project, makes it the ideal choice for large-scale applications like review analysis and customer feedback systems.

While both models serve distinct purposes, the LSTM model demonstrated better overall performance and adaptability, particularly for tasks requiring deeper language understanding. However, the simplicity and speed of Naive Bayes still make it a practical baseline for many text classification challenges.

3.2 Project Summary and Reflections

This assignment turned out to be a great learning opportunity. I gained substantial knowledge on the pros and cons of the statistical and embedding based techniques for undertaking text classification tasks. The application of both Naive Bayes and LSTM models made me realize the relevance of data preprocessing, model structure, and evaluation metrics in achieving adequate outcomes.

The use of the IMDb dataset brought in quite a few challenges such as coping with dirty data and finding a trade-off between a basic model and a complicated one. Naive Bayes started off as a simple baseline but LSTM managed to incorporate semantic and contextual information and improved the performance of the classification greatly.

There are different situations that these models fit to in real life. For example, LSTM is not practical for spam filtering since it requires constant results that can be easily explained especially when the resources are limited, but with a customer feedback tool it excels.

Improvements That Can Be Made:

There were some measures that were noted for improvement during the course of the project.

- Some of the other embeddings that could be added are GloVe and Word2Vec embedded with LSTM and this could improve the performance significantly. This is because it will exploit the already known semantics of the words.
- Some advanced algorithms could also be included and they include early stopping, dropout tuning hyperparameter optimization, to avoid overfitting and make the LSTM model stronger.
- Further, some hybrid models could also be tested in performance whether it be: incorporating TF-IDF features with LSTM or any other hybrid model as such to assure both speed and proper handling of context.

New Directions on Research:

Some paths can be built on the current analysis which include the following:

- Other multilingual datasets can also be evaluated to help in the analysis expanding it to the single one we incorporated in the current analysis.
- Further, these attention mechanisms for example could not satisfy the needs of deep learning models, which demanded for these to be explainable, but the urge for models to be understandable is constantly growing.
- In addition, other transformer based models for example BERT could be evaluated instead on the large and diverse datasets.

Thus, it has deepened understanding of text classification and highlighted the trade-offs between traditional statistical models and advanced deep learning methods. By addressing the suggested improvements and exploring future research directions, these models can be further refined for real-world applications in sentiment analysis and beyond.

References

1. Levity.ai (2024) *Text classifiers in machine learning: A practical guide*. Available at: <https://levity.ai/blog/text-classifiers-in-machine-learning-a-practical-guide> (Accessed: 5 January 2025).
2. Dataiku (2024) *7 text classification techniques for any scenario*. Available at: <https://blog.dataiku.com/7-text-classification-techniques-for-any-scenario> (Accessed: 5 January 2025).
3. Nikmah, T.L., Ammar, M.Z., Allatif, Y.R., Husna, R.M.P.A., Kurniasari, P.A., and Bahri, A.S. (2022) 'Comparison of LSTM, SVM, and naive bayes for classifying sexual harassment tweets', *Journal of Soft Computing Explorations*, 3(2), pp. 131-137. Available at: <https://shmpublisher.com/index.php/joscex/article/download/85/64> (Accessed: 5 January 2025).
4. ActiveWizards (2024) 'Sentiment Analysis with Naive Bayes and LSTM', *ActiveWizards Blog*. Available at: <https://activewizards.com/blog/sentiment-analysis-with-naive-bayes-and-lstm/> (Accessed: 5 January 2025).
5. Towards Data Science (2020) 'Naive Bayes and LSTM Based Classifier Models', *Towards Data Science*. Available at: <https://towardsdatascience.com/naive-bayes-and-lstm-based-classifier-models-63d521a48c20> (Accessed: 5 January 2025).
6. Lakshmi, N. (2018). *IMDB Dataset of 50K Movie Reviews*. [online] Kaggle. Available at: <https://www.kaggle.com/datasets/lakshmi25npathi/imdb-dataset-of-50k-movie-reviews/data> [Accessed 5 Jan. 2025].

Appendix

Some Code Snippets

```
[ ] from sklearn.model_selection import train_test_split

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(
    data['review'], # Features (reviews)
    data['sentiment'], # Labels (sentiment)
    test_size=0.2, # 20% for testing
    random_state=42 # For reproducibility
)

# Display the sizes of the splits
print("Training set size:", len(X_train))
print("Testing set size:", len(X_test))
```

Training set size: 40000
Testing set size: 10000

Figure 9: Code Snippet for Data Splitting – Demonstrates splitting the dataset into training and testing sets using an 80-20 split

```
[ ] from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import classification_report

# TF-IDF Vectorization
vectorizer = TfidfVectorizer(max_features=5000) # Limit to 5000 features for simplicity
X_train_tfidf = vectorizer.fit_transform(X_train) # Fit and transform the training data
X_test_tfidf = vectorizer.transform(X_test) # Transform the testing data

# Train a Naive Bayes classifier
model_nb = MultinomialNB()
model_nb.fit(X_train_tfidf, y_train)

# Make predictions
y_pred = model_nb.predict(X_test_tfidf)

# Evaluate the model
print(classification_report(y_test, y_pred))
```

	precision	recall	f1-score	support
negative	0.85	0.85	0.85	4961
positive	0.85	0.85	0.85	5039
accuracy			0.85	10000
macro avg	0.85	0.85	0.85	10000
weighted avg	0.85	0.85	0.85	10000

Figure 10: Implementation and Evaluation of the Naive Bayes Classifier

```
[ ] from tensorflow.keras.preprocessing.text import Tokenizer
    from tensorflow.keras.preprocessing.sequence import pad_sequences

    # Parameters for tokenization
    vocab_size = 5000 # Maximum number of words to keep
    max_length = 200 # Maximum review length in words
    embedding_dim = 128 # Embedding dimension

    # Tokenize the text
    tokenizer = Tokenizer(num_words=vocab_size, oov_token="<OOV>")
    tokenizer.fit_on_texts(X_train)

    # Convert text to sequences
    X_train_seq = tokenizer.texts_to_sequences(X_train)
    X_test_seq = tokenizer.texts_to_sequences(X_test)

    # Pad sequences to ensure uniform length
    X_train_padded = pad_sequences(X_train_seq, maxlen=max_length, padding='post', truncating='post')
    X_test_padded = pad_sequences(X_test_seq, maxlen=max_length, padding='post', truncating='post')

    # Display the shape of the padded data
    print("Padded training data shape:", X_train_padded.shape)
    print("Padded testing data shape:", X_test_padded.shape)
```


 Padded training data shape: (40000, 200)
 Padded testing data shape: (10000, 200)

Figure 11: Tokenization and Padding for LSTM Model

```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/core/embedding.py:90: UserWarning: Argument `input_length` is deprecated. Just remove it.
  warnings.warn(
Model: "sequential"
```

Layer (type)	Output Shape	Param #
embedding (Embedding)	?	0 (unbuilt)
lstm (LSTM)	?	0 (unbuilt)
dropout (Dropout)	?	0 (unbuilt)
lstm_1 (LSTM)	?	0 (unbuilt)
dense (Dense)	?	0 (unbuilt)

```

Total params: 0 (0.00 B)
Trainable params: 0 (0.00 B)
Non-trainable params: 0 (0.00 B)
Encoded labels example: [0 0 1 0 0]
Epoch 1/5
1000/1000 ————— 517s 514ms/step - accuracy: 0.4991 - loss: 0.6937 - val_accuracy: 0.5175 - val_loss: 0.6916
Epoch 2/5
1000/1000 ————— 568s 520ms/step - accuracy: 0.5252 - loss: 0.6873 - val_accuracy: 0.5297 - val_loss: 0.6818
Epoch 3/5
1000/1000 ————— 557s 515ms/step - accuracy: 0.7515 - loss: 0.4989 - val_accuracy: 0.8649 - val_loss: 0.3170
Epoch 4/5
1000/1000 ————— 560s 513ms/step - accuracy: 0.8998 - loss: 0.2579 - val_accuracy: 0.8704 - val_loss: 0.3059
Epoch 5/5
1000/1000 ————— 564s 515ms/step - accuracy: 0.9202 - loss: 0.2069 - val_accuracy: 0.8731 - val_loss: 0.3081
```

Figure 12: LSTM Model Architecture and Training

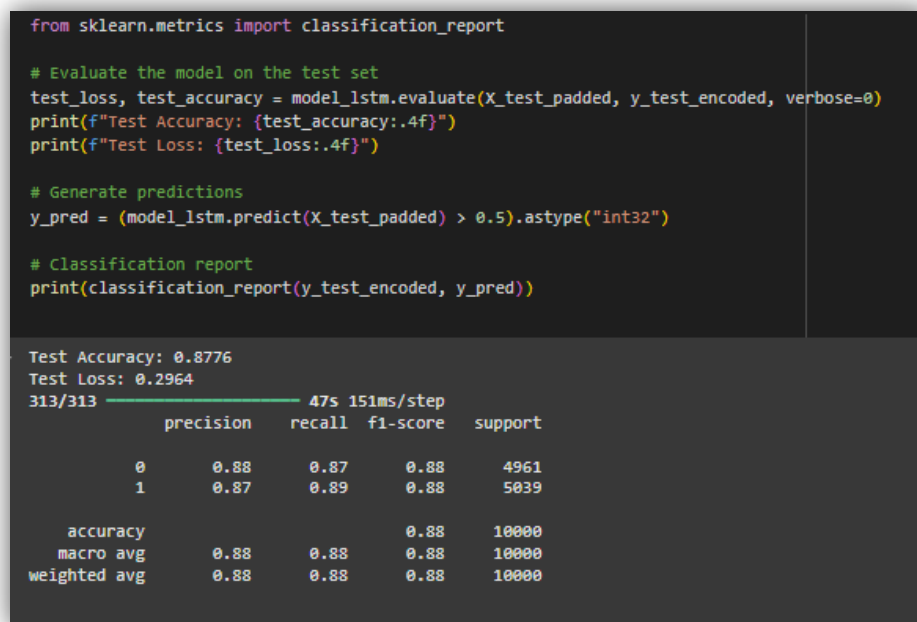


Figure 13: LSTM Model Evaluation

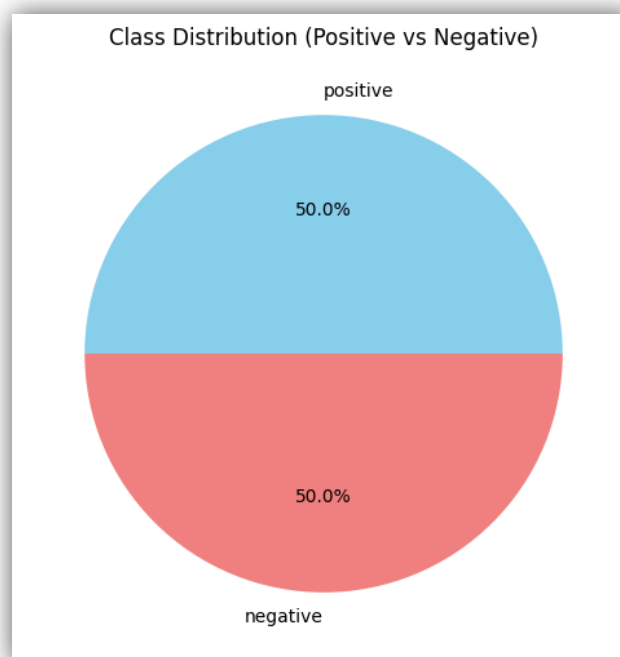


Figure 14: Class Distribution of IMDb Dataset