Sentiment Analysis

User Reviews using NLP and Deep Learning

High-Level Project Overview:

This project aims to analyse 8,000 user reviews—4,000 positive and 4,000 negative—to automatically classify their sentiment. The workflow comprises several steps, including data pre-processing, feature extraction, model selection, hyperparameter tuning, and model evaluation. We leverage Natural Language Processing (NLP) techniques and Deep Learning models to achieve robust and accurate sentiment classification.

Data Pre-processing

Data Preparation

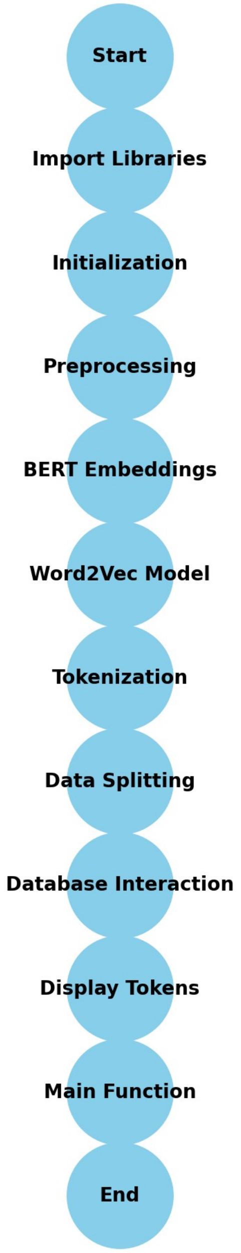
Model Building

Choose between **Flatten-based** or **LSTM models**.

Hyperparameter Tuning.

Model Training

Model Evaluation.



STEP 1

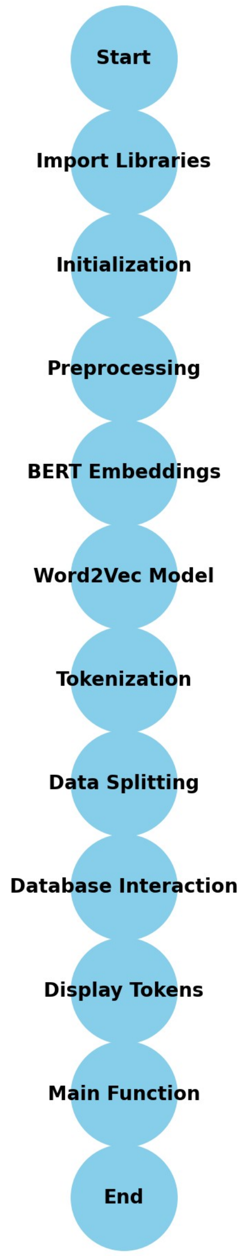
Pre-processing and tokenization:

The goal of this project is to perform sentiment analysis on a dataset of 8,000 reviews, evenly distributed between positive and negative sentiments. Proper pre-processing and tokenization of the text data are crucial steps to ensure the model can accurately capture the nuances in the reviews.

Introduction:
The chosen libraries and resources are essential for various stages of text data pre-processing and model training. NLTK is used for tokenization, POS tagging, and chunking, which are necessary for preparing the text data. **BeautifulSoup** is used for extracting reviews from an XML file, while **Gensim** is used for training a Word2Vec model to generate word embeddings that capture semantic relationships between words. Scikit-learn is used for data splitting, and additional resources like **stopwords**, **RegexpTokenizer**, **PorterStemmer**, and **WordNetLemmatizer** are used for text cleaning and normalization.

Reasoning:
The dataset is large and diverse, with a vocabulary of 49,220 unique tokens and reviews of varying lengths. This necessitates reducing data dimensionality and focusing on the most informative words and phrases. Additionally, the nuanced usage of words like 'good' and the recurrence of words like 'time' across both positive and negative reviews indicate that a simple bag-of-words approach may not suffice. Therefore, word embeddings, which capture semantic relationships, are essential for accurate sentiment analysis.

The selected libraries and resources enable comprehensive text data preprocessing and ensure the model accurately captures the nuances in the reviews for sentiment analysis. This approach is vital for handling the nuanced usage of words and ensuring a more accurate and nuanced sentiment analysis.



Pre-processing and Analysis Steps:

1.**Data Extraction**: Extract the review texts from an XML file using the **extract_review_texts_from_file** function.

2.**Handle Contractions**: Expand contractions in the review texts using the **handle_contractions** function.

3.**Remove Hyperlinks, Mentions, and Hashtags**: Remove hyperlinks, mentions, and hashtags from the review texts using regular expressions.

4.**Convert to Lowercase**: Convert all the text to lowercase.

5.**Remove HTML Tags**: Remove HTML tags from the review texts using regular expressions.

6.**Tokenization**: Tokenize the review texts into words using the **RegexpTokenizer** from **nltk**.

7.**Remove Punctuation**: Remove punctuation and unwanted characters/sequences from the tokens.

8.**Remove Numbers**: Remove numbers from the tokens.

9.**Remove Stopwords**: Remove common English stopwords and category-specific stopwords from the tokens.

10.**Part-of-Speech Tagging**: Perform part-of-speech tagging on the tokens using **nltk.pos_tag**.

11.***Chunking**: Perform chunking on the POS-tagged tokens using a predefined chunk grammar and **nltk.RegexpParser**.

12.**Stemming**: Perform stemming on the tokens using the **PorterStemmer** from **nltk**.

13.**Lemmatization**: Perform lemmatization on the tokens using the **WordNetLemmatizer** from **nltk**.

14.**Frequency Distribution**: Compute the frequency distribution of the tokens and chunked phrases using **FreqDist** from **nltk**.

15.***BERT Tokenization**: Tokenize the processed reviews using BERT's tokenizer.

16.**Data Splitting**: Split the processed and tokenized data into training, validation, and test sets.

17.**Database Selection**: Select a database to add the tokens.

18.**Add Tokens to Database**: Add the tokens to the selected database using the **add_tokens_to_database** function.

19.**View Token Report**: View a report on the tokens stored in the database using the **view_tokens_report** function.

20.**Train Word2Vec Model**: Train a Word2Vec model on the processed reviews using the **train_word2vec** function.

Challenges

Chunking

Chunking involves extracting meaningful phrases from unstructured text and is crucial for text processing applications like information extraction and named entity recognition.

Initial Challenges

The initial chunking process produced incomprehensible results, such as **designed_vertical_cd_rack_doesnt_individual_slots_cds**, which is not meaningful or useful for subsequent analysis.

Solution

The grammar rules for chunking were fine-tuned and specific patterns for noun phrases, verb phrases, and negative adjectives were defined:

```
chunk_grammar = r"""
  NP: {<DT|JJ>*<NN.*>+}
  VP: {<VB.*><NP|PP|RB>*}
  NegAdj: {<RB><JJ>}
"""
```

Additionally, chunks were filtered based on their length to remove very short or very long chunks.

Result

The fine-tuning and filtering resulted in clear and meaningful chunks, such as **easy_use** , **sound_quality**, and **good_price**, which are much more informative and useful for subsequent analysis.

Careful definition of grammar rules and filtering criteria are crucial for meaningful chunking. The fine-tuned rules and criteria significantly improved the chunking process, resulting in clear and meaningful chunks of information.

Exploration of BERT Integration

We explored the integration of BERT for tokenization and obtaining sentence embeddings.
Used BertTokenizer and BertModel from the transformers library and get_bert_embeddings function to convert text into tokens and then into embeddings.
Developed a bert_tokenize function to convert data into input ids, attention masks, and token type ids for model input.

Data Pre-processing and Tokenization

Overview for Sentiment Analysis Mode

The dataset encompasses **8,000** reviews, evenly distributed between positive and negative sentiments. Post-tokenization, positive reviews generated **923,804** tokens, with **115,696** being unique, while negative reviews yielded 893,861 tokens, of which **109,788** are distinct. This slight variance in vocabulary richness suggests a modestly more diverse lexicon in positive reviews. The entire corpus boasts a vocabulary size of **49,220** unique tokens.

For analytical consistency, reviews were padded to equate the length of the longest review at **7,646** tokens. On average, negative reviews (**223.47** tokens) are somewhat more concise than positive ones (**230.95** tokens).

The prominence of **'great'** as a leading token in positive reviews signifies its potential as a cornerstone for enhanced sentiment analysis.

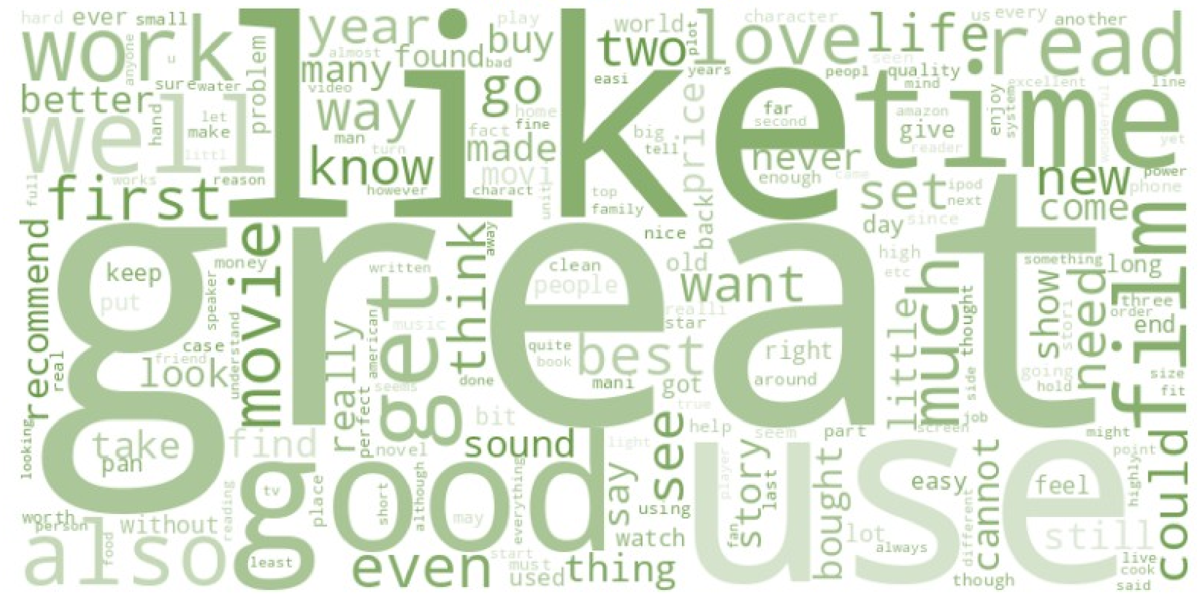
Intriguingly, the word **'good'** appears in both positive and negative reviews, underscoring its nuanced usage. Moreover, the token '**time**' is recurrent across both sentiments.

Data Overview:	**Positive Reviews**	**Negative Reviews**
Loaded Reviews	4000	4000
Total Tokens	923,804	893,861
Unique Tokens	115,696	109,788
Average Length of Reviews	230.95 tokens	223.47 tokens

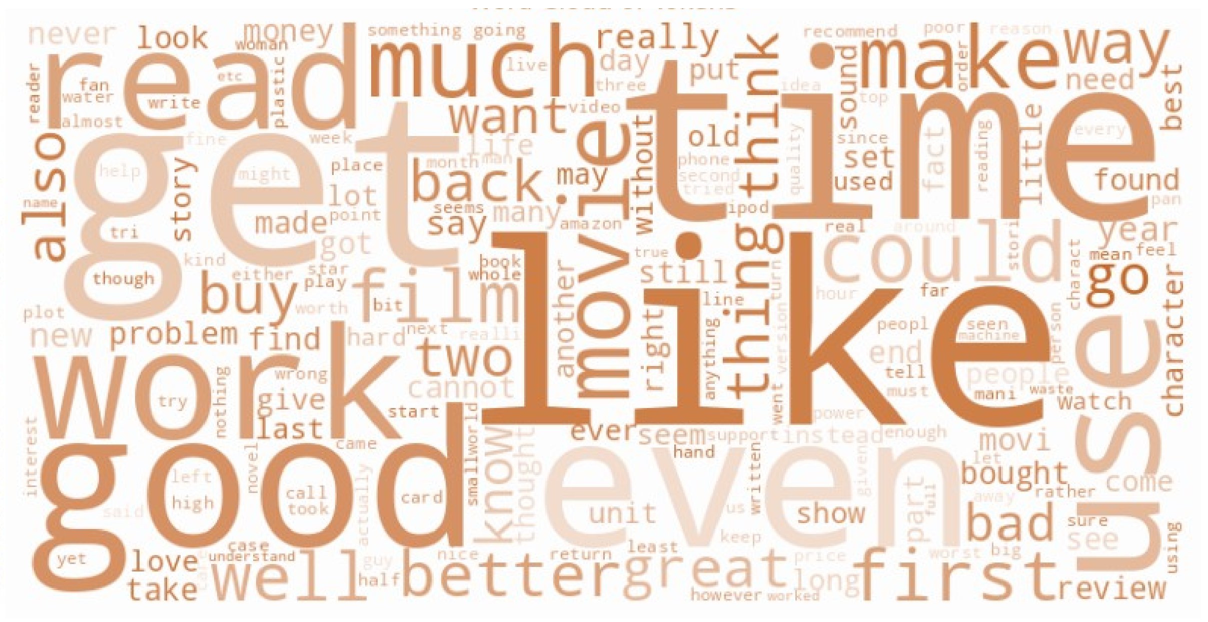
Combined Reviews	8000
Label Distribution (Negative)	4000
Label Distribution (Positive)	4000
Total Vocabulary Size	49,220
Maximum Review Length	7,646 tokens
Most Frequent Token	“time” x12,418

Sentimental Lexicon:
Visualizing Prominent Review Tokens through Hierarchical Word Cloud Analysis

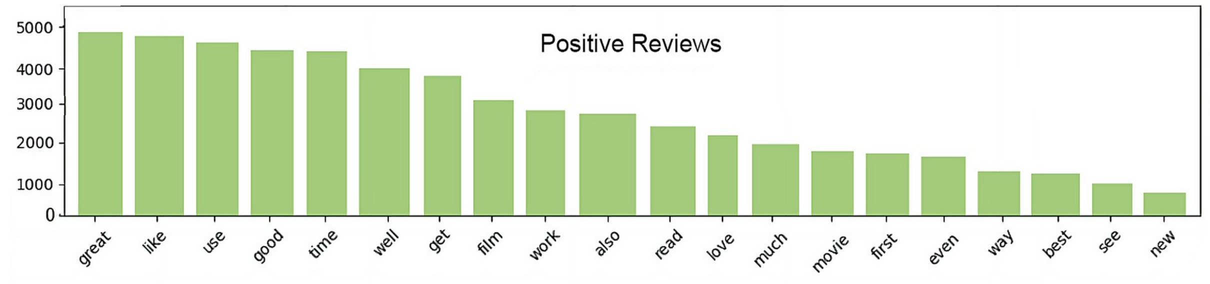
Positive Reviews

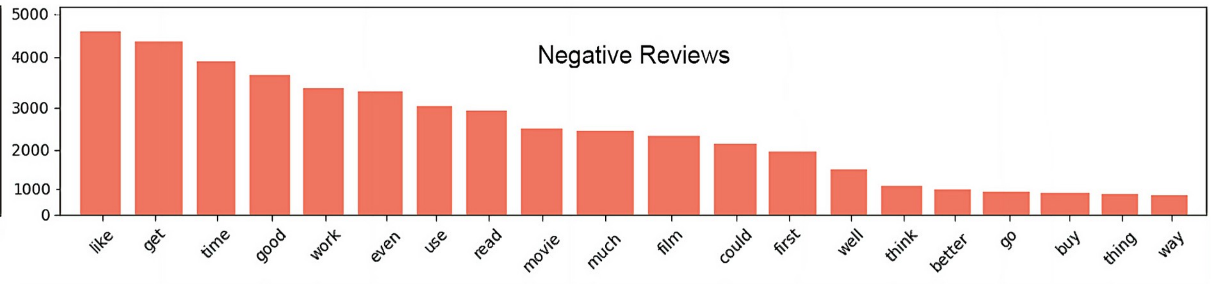


Negative Reviews

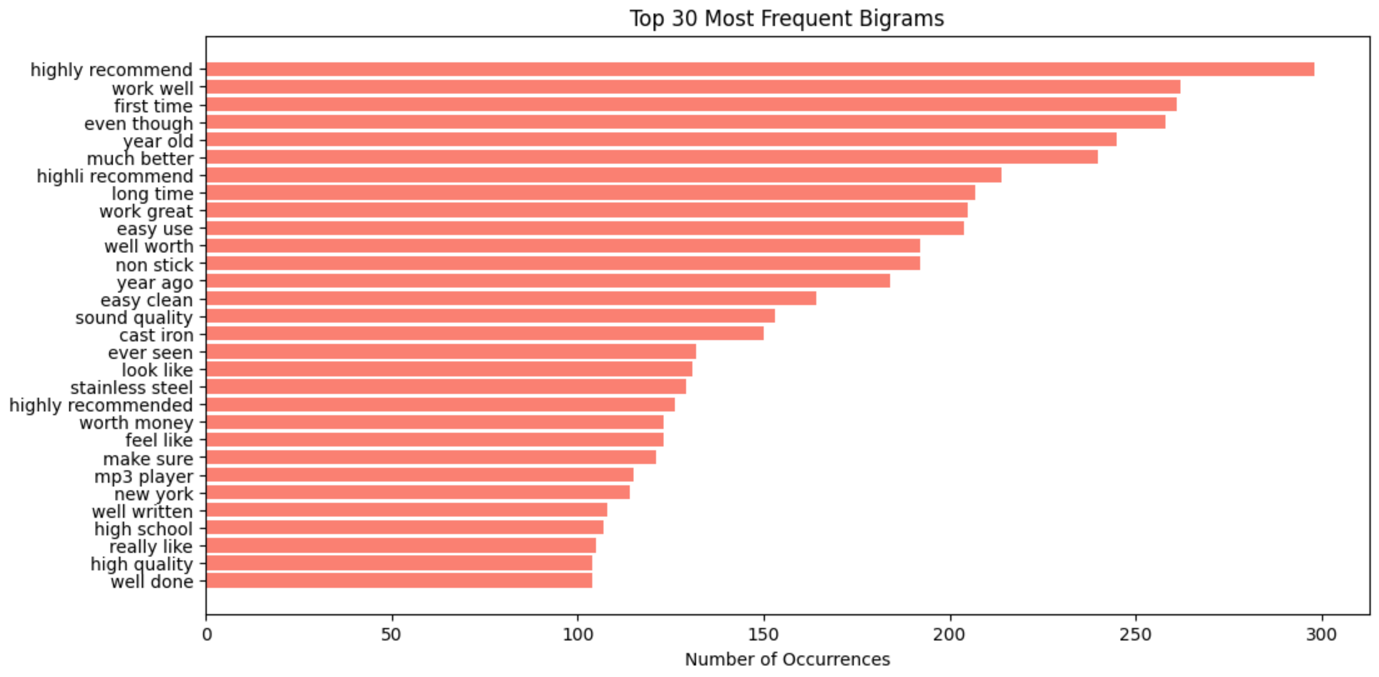


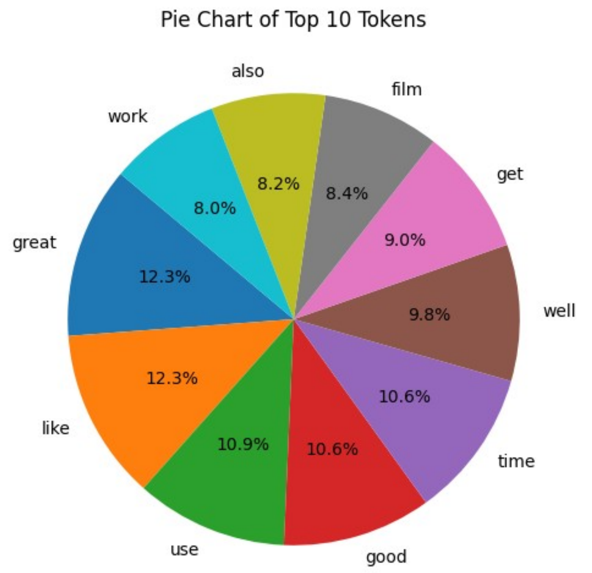
Top 20 tokens by Frequency



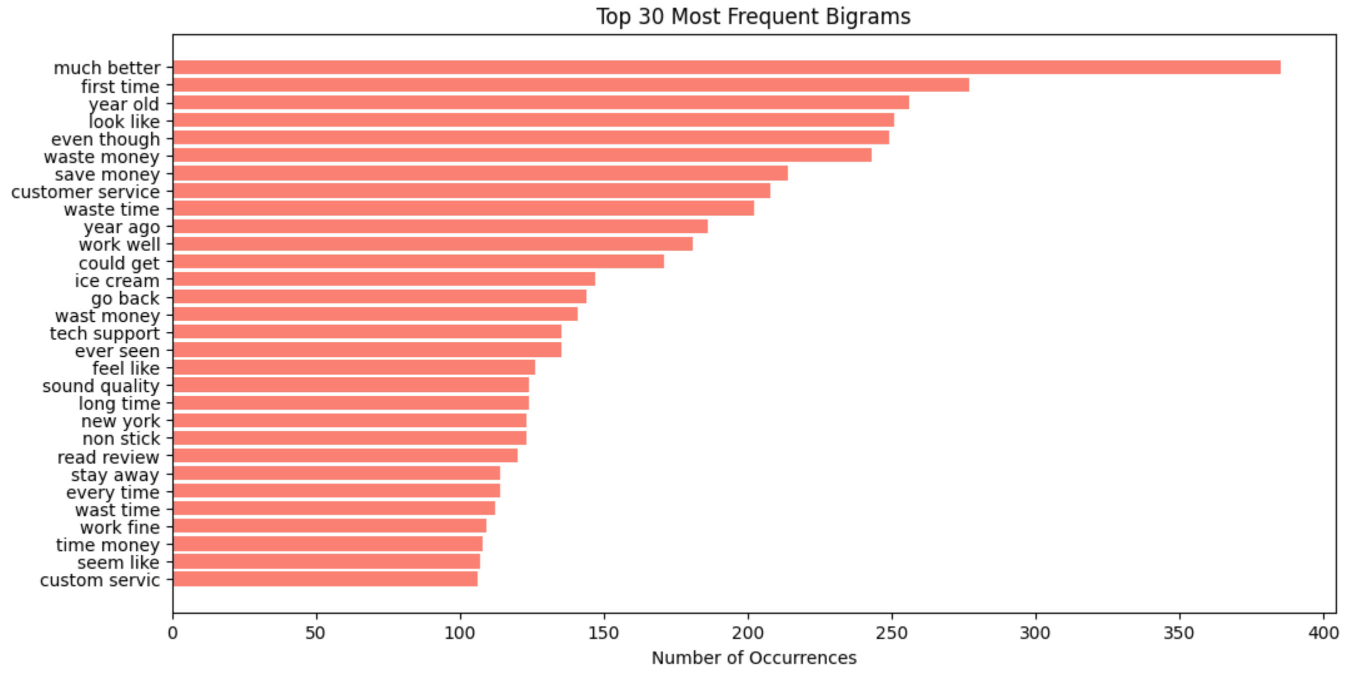


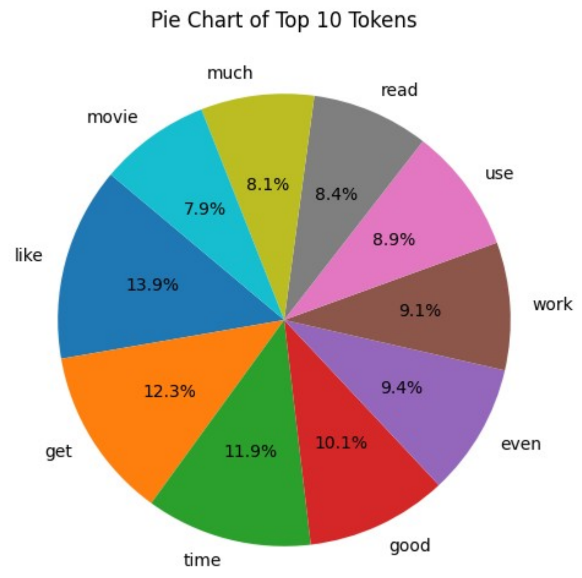
Positive Reviews:





Negative Reviews:



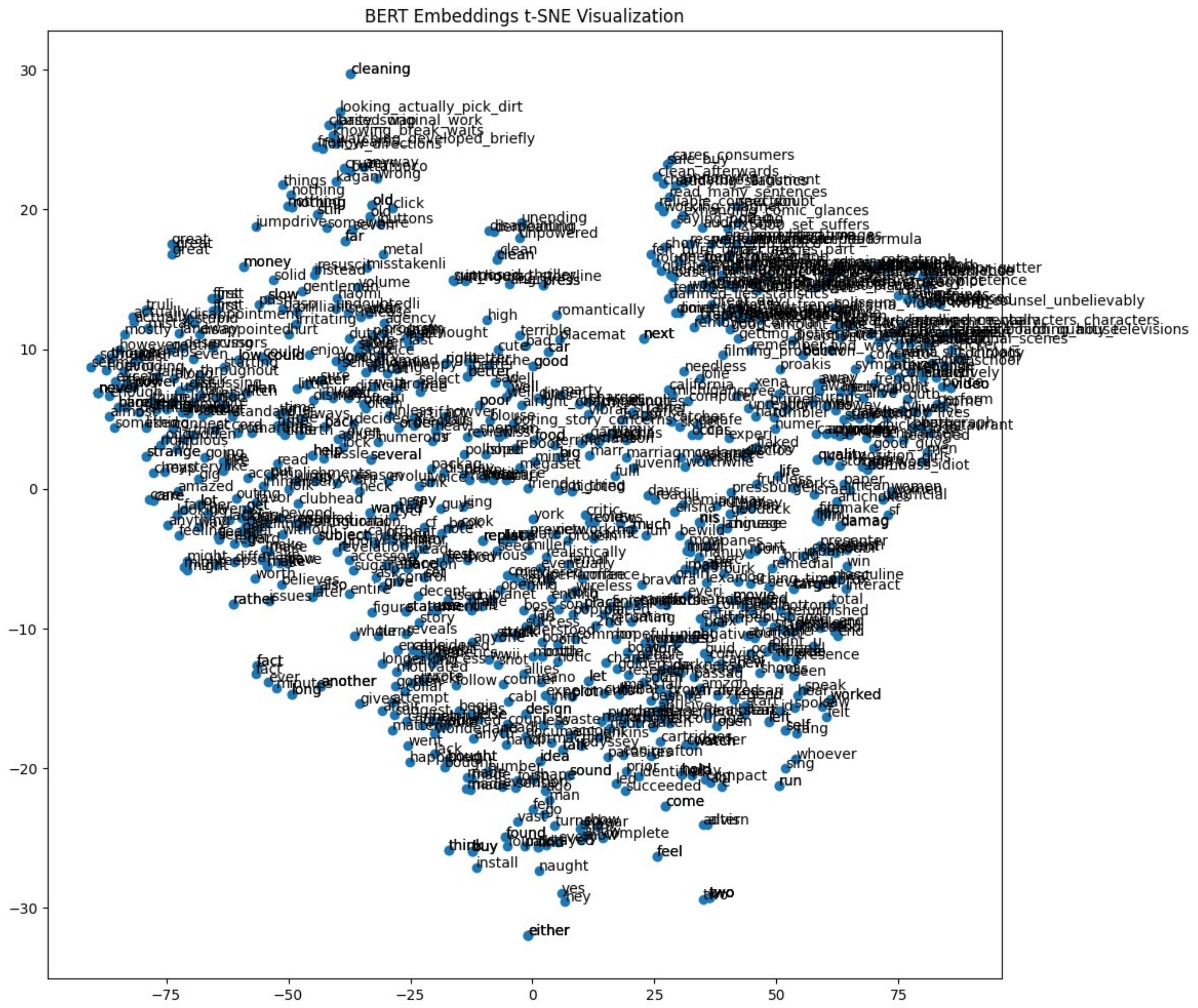


BERT Embedding t-SNE Visualization:

Quick Overview

BERT: A pre-trained language model that understands text context.
t-SNE: A visualization technique that reduces data dimensions.
Embedding: Conversion of words into numerical vectors.

What the Graph Shows:
Clusters: Groups of similar words or sentences.
Outliers: Unique or rare words.
The graph visualizes how closely related different words or sentences are, based on their BERT embeddings.



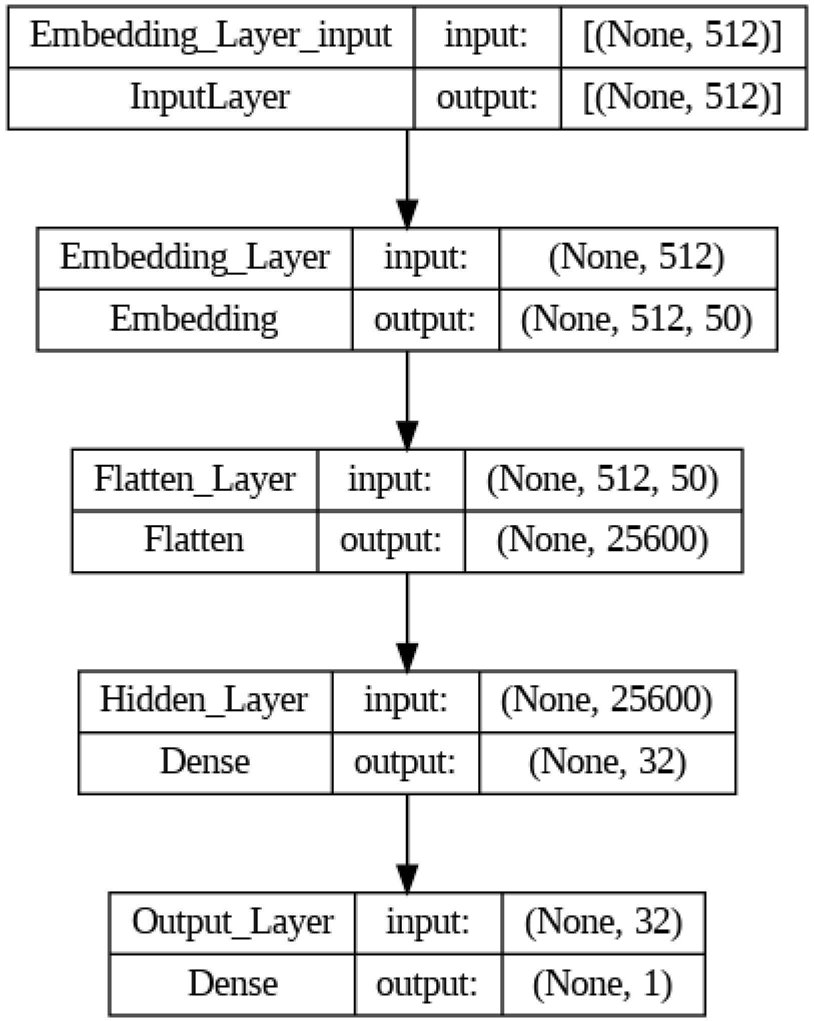
Model Construction and Hyperparameter Tuning for Sentiment Analysis

High-Level Overview:

This step focuses on defining, building, and tuning the machine learning models for sentiment analysis. It involves setting up the architecture for two types of neural networks: a simpler Flatten-based model and a more complex Long Short-Term Memory (LSTM) model. Additionally, it incorporates hyperparameter tuning using Keras Tuner to optimize the models for better performance.

Key Elements:

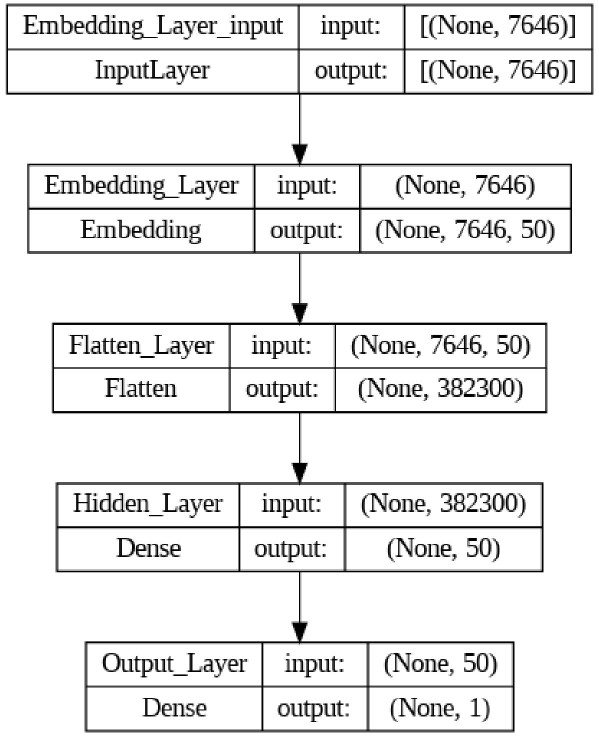
Library Imports: Import TensorFlow, **Keras**, and **Keras Tuner** libraries for model construction and tuning.
Session Reset: Clear any previous TensorFlow sessions to start fresh.
Set Seed: Set a random seed for reproducible results.
Hyperparameters: Define initial hyperparameters like embedding dimensions, hidden units, learning rate, etc.
Model Definitions: Function-based architecture for Flatten and LSTM models.
Callbacks: Utilize Early Stopping and Learning Rate Reduction for efficient training.
Hyperparameter Tuning: Employ Keras Tuner for finding the optimal model configurations.
User Interface: Interactive dropdown and button widgets for model selection.



ANN Architectural exploration

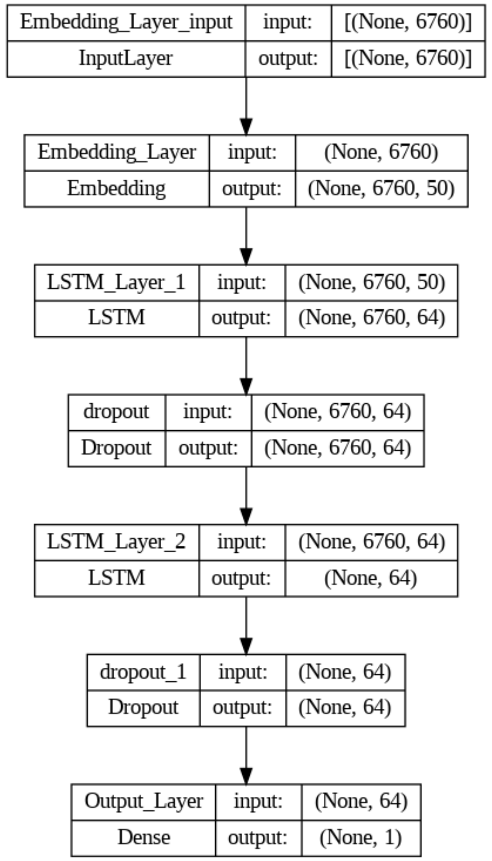
Model: Flatten-Based Architecture.

An embedding layer to handle word embeddings. A flatten layer to flatten the sequential data, making it suitable for the dense layers. A dense hidden layer with a ReLU activation. An output dense layer with a sigmoid activation for binary classification. This architecture is straightforward and might work well for simpler datasets. However, the flatten layer doesn't capture the sequential nature of text data, which is why we explore an alternative architecture.


Embedding_Layer_input | input: | [(None, 7646)]
InputLayer | output: | [(None, 7646)]
Embedding_Layer | input: | (None, 7646)
Embedding | output: | (None, 7646, 50)
Flatten_Layer | input: | (None, 7646, 50)
Flatten | output: | (None, 382300)
Hidden_Layer | input: | (None, 382300)
Dense | output: | (None, 50)
Output_Layer | input: | (None, 50)
Dense | output: | (None, 1)

Model: LSTM-Based Architecture.

An embedding layer to handle word embeddings. An LSTM (Long Short-Term Memory) layer, which is a type of recurrent neural network (RNN) layer. RNNs are designed to work with sequence data like text, and LSTM is a specific variant that can capture long-term dependencies in the data. A dense hidden layer with a ReLU activation. An output dense layer with a sigmoid activation for binary classification. This architecture is more complex but has the potential to perform better on text data due to the LSTM layer's ability to understand sequences and capture temporal dynamics.


Embedding_Layer_input | input: | [(None, 6760)]
InputLayer | output: | [(None, 6760)]
Embedding_Layer | input: | (None, 6760)
Embedding | output: | (None, 6760, 50)
LSTM_Layer_1 | input: | (None, 6760, 50)
LSTM | output: | (None, 6760, 64)
dropout | input: | (None, 6760, 64)
Dropout | output: | (None, 6760, 64)
LSTM_Layer_2 | input: | (None, 6760, 64)
LSTM | output: | (None, 64)
dropout_1 | input: | (None, 64)
Dropout | output: | (None, 64)
Output_Layer | input: | (None, 64)
Dense | output: | (None, 1)

Data Pre-processing and Model Training Workflow

High-Level Overview:

This step is geared towards creating an end-to-end workflow for model training and evaluation. It integrates a user interface for interactive model selection and incorporates data management utilities for a streamlined experience.

Key Elements:

Library Imports: Incorporate necessary libraries for array manipulations, file operations, and user interface.
Clear Previous Data: Functionality to remove old hyperparameter tuning data.
Set Save Directory: Specify the directory where trained models and their histories will be saved.
User Interface: Interactive widgets for model naming and triggering the training process.
Data Metrics Display: Show relevant statistics related to the dataset, such as vocabulary size and average review length.
Training Configuration: Utilize Keras Tuner and callbacks for hyperparameter tuning and efficient training.
Model Saving: Save the best-performing model and its training history for future use.
This step encapsulates the entire training workflow, providing a platform for user interaction and model performance tuning. It ensures that the best model is saved for future evaluations and applications.

Challenges

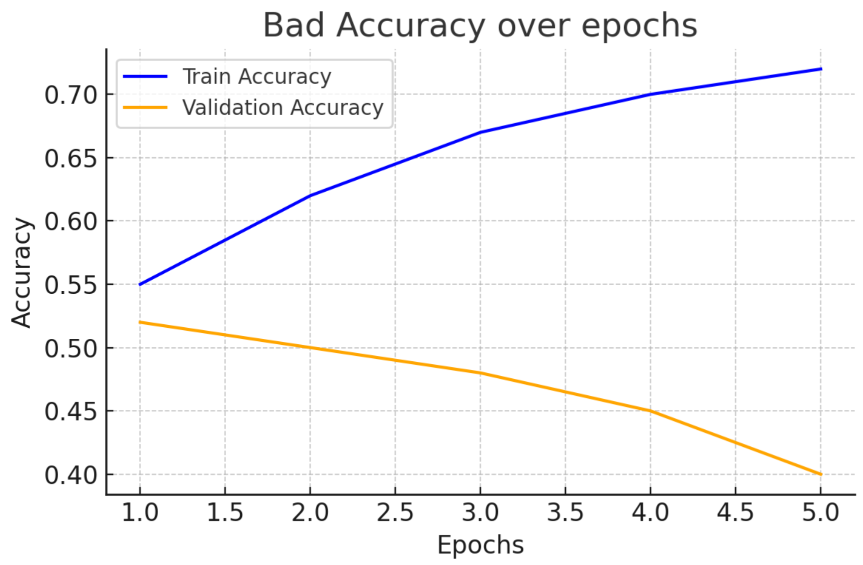
Hyperparameter Sensitivity and Tuning with Keras Tuner

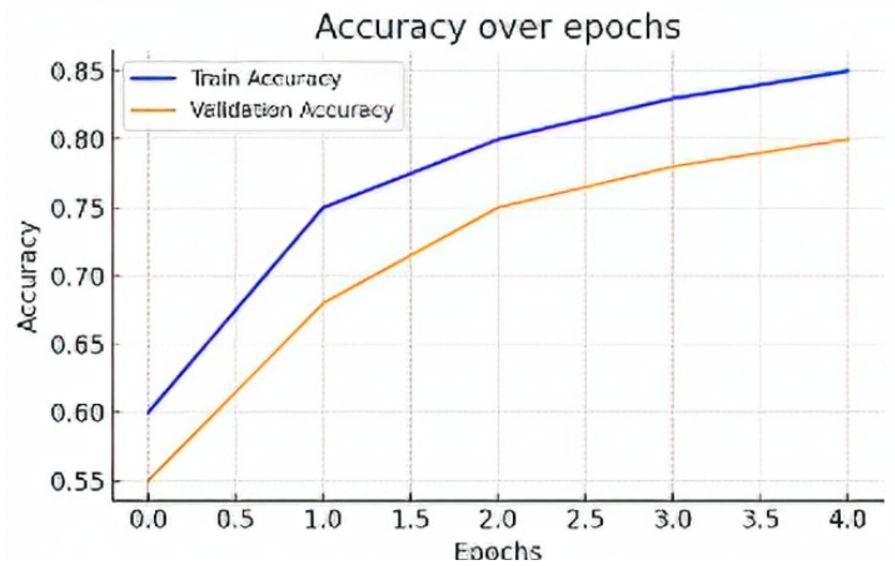
Computational Cost:
 Keras Tuner makes multiple training runs, expensive in time and resources.
Risk of Overfitting: Optimizing for a validation set may not generalize well.
Local Minima: May find only a 'good enough' solution, not the best.
Sensitivity:
Learning Rate: Critical for model performance.
Batch Size: Affects accuracy and computational cost.
Early Stopping: Stops training at the right time, if well-tuned.
Keras Tuner:
Used Keras Tuner (kt.Hyperband) for automated hyperparameter optimization.
Clears old tuning data for a fresh run.
Implements early stopping and learning rate reduction for efficient tuning.
Conclusion:
Hyperparameter tuning is crucial but comes with challenges. Using Keras Tuner as in your code helps automate this process and adopts good practices to mitigate risks.

Model Interpretation:

Accuracy Over epoch

This graph shows how the model's accuracy progresses over each training epoch for both the training data and the validation data. The horizontal axis represents the epochs (iterations over the entire dataset), and the vertical axis represents accuracy.





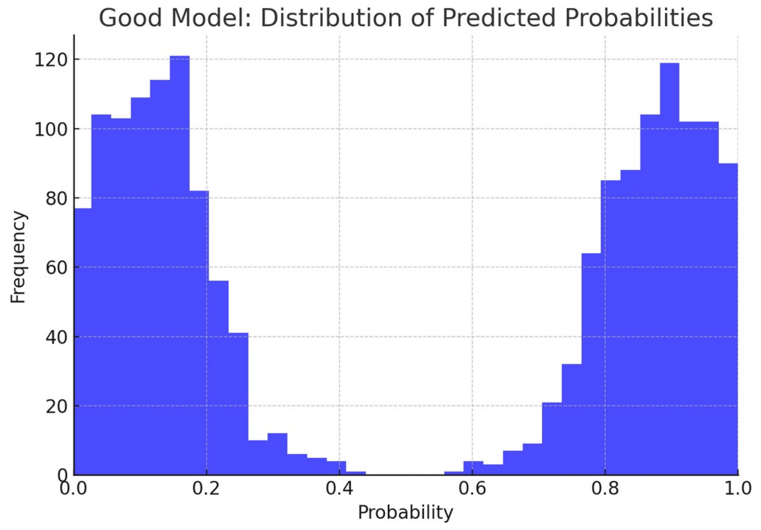
Example of a good "Accuracy over epochs" graph:
Explanation:
Convergence: Both training and validation accuracy increase over time, showing that the model is learning.
Stability: The curves start to plateau towards the later epochs, suggesting that the model is reaching its optimal performance on the current data and architecture.
Gap between training and validation: The gap between the training and validation accuracy is small, indicating that the model is not overfitting. Overfitting happens when a model performs well on the training data but poorly on unseen data (like the validation set).

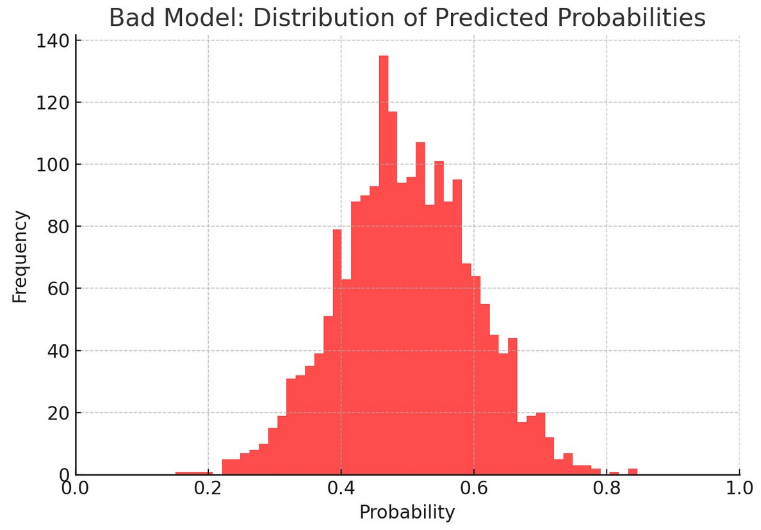
Example of a bad "Accuracy over epochs" graph:
Explanation:
1. No Proper Convergence: While the training accuracy is slightly increasing, the validation accuracy is decreasing. This indicates that the model might be overfitting, meaning it is getting better at the training data but performing worse on new, unseen data.
2. Increasing Gap: The increasing gap between the training and validation accuracy further confirms overfitting. A widening gap as epochs progress is a classic sign of overfitting.
Low Overall Accuracy: Both the training and validation accuracies are relatively low, suggesting that the model isn't learning effectively from the data. This could be due to various reasons like inadequate model architecture, poor data quality, or incorrect hyperparameters.

Model Interpretation:

Distribution of Predicted Probabilities

This graph shows the distribution of the probabilities predicted by the model for the positive class. The x-axis represents the predicted probability values (ranging from 0 to 1), and the y-axis represents the frequency (or density) of instances that have those probabilities.

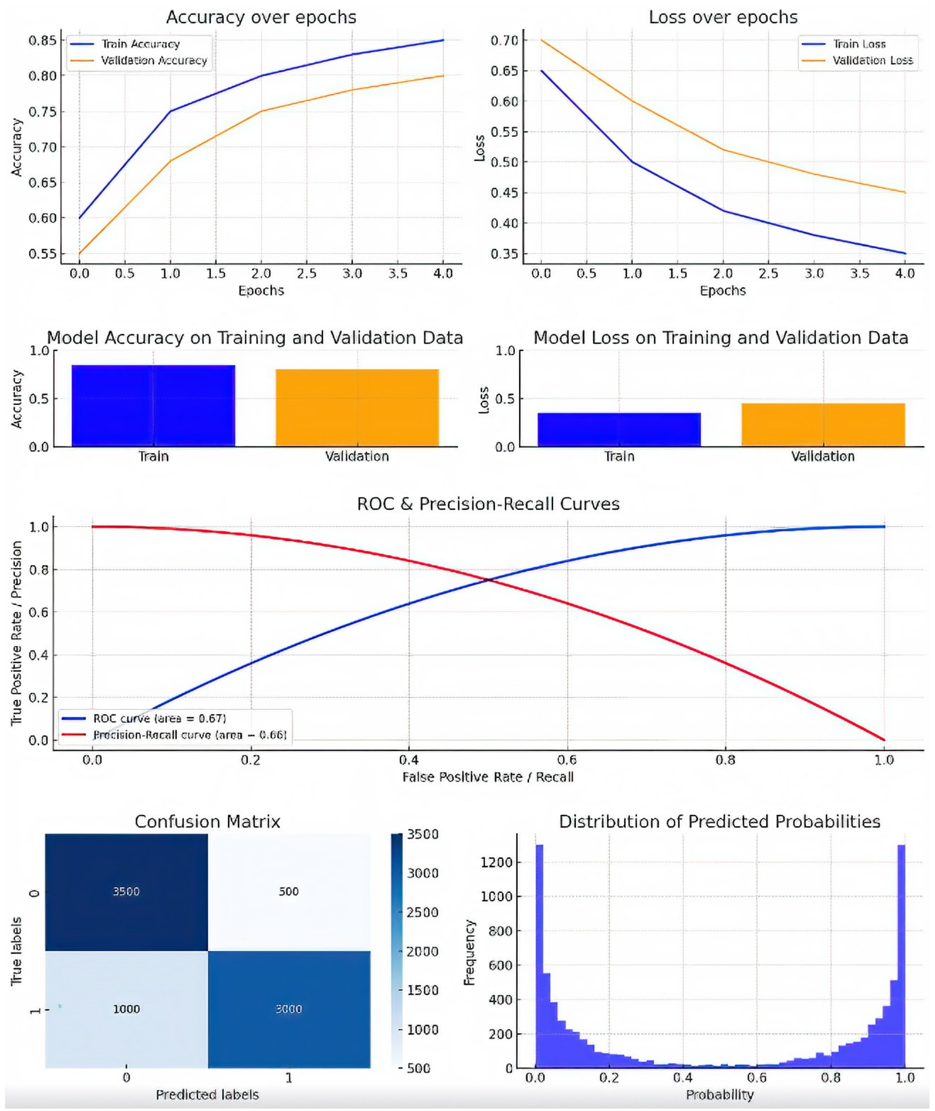




Good Model
• The distribution has two clear peaks, one near 0 and the other near 1. This indicates that the model is very confident in its predictions.
• Most predictions are concentrated at the extremes, suggesting that the model can distinguish between positive and negative instances effectively.
Bad Model (Right side):

Bad Model
• The distribution is concentrated around the 0.5 mark. This shows that the model is uncertain about a large portion of its predictions and is essentially guessing.
• A lack of clear peaks at the extremes suggests the model struggles to confidently classify instances into either the positive or negative class.
The distribution of predicted probabilities can give insights into how decisive or "confident" a model is in its predictions. Ideally, we want a model that can confidently assign probabilities close to 0 or 1, rather than being uncertain and assigning values around 0.5.

Ideal Model



Our Model

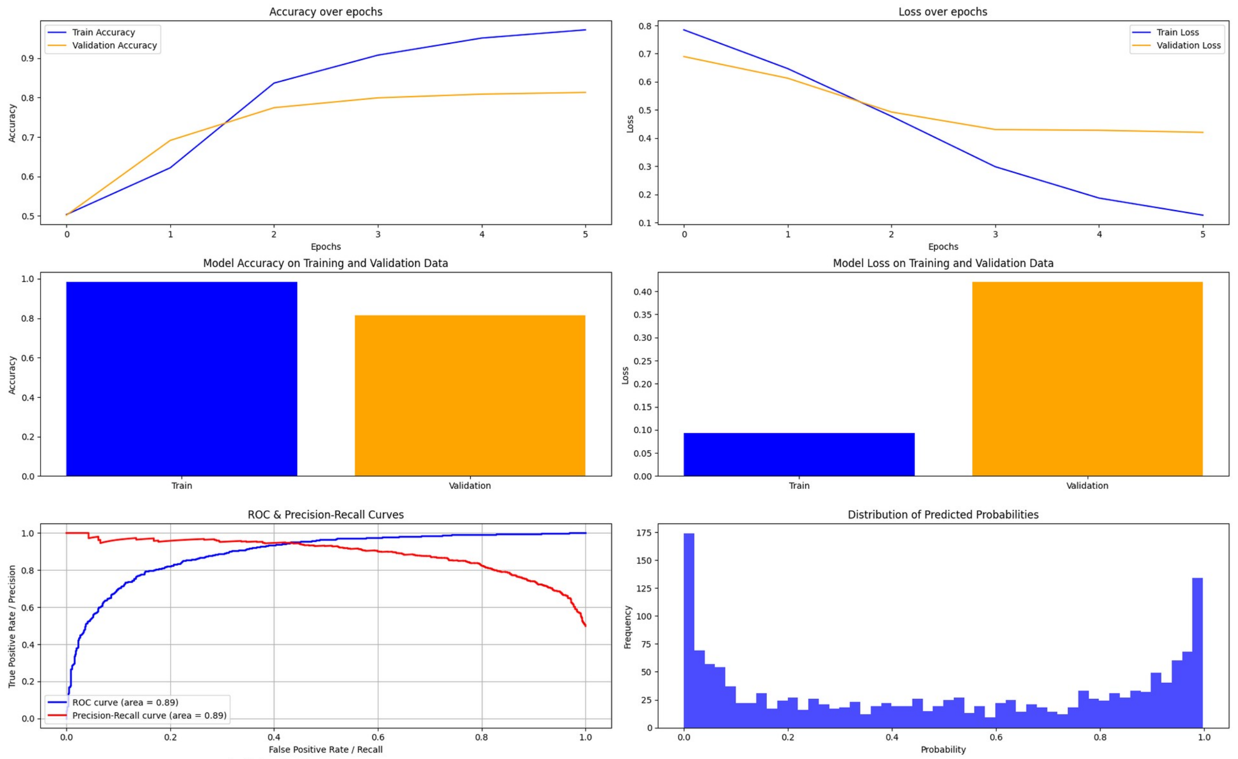
Model: Model_015

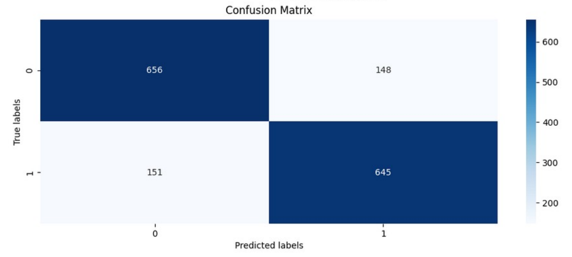
Model Size: 246.95 MB

Total Parameters: 21576101

Hyperparameters

Embedding Dimension: 50
Dense (hidden) Units: 50
Dense (hidden) Activation: relu
Dense (output) Units: 1
Dense (output) Activation: sigmoid
Learning Rate: 0.0005
Test Size: 0.2
Epochs: 6
Batch Size: 128





Similar Words Research

great:	wonderful, amaz, highly, easy, delicious, son, perfect, solid
perfect:	excellent, solid, awesome, fantast, best, amazing, glad, highli
happy:	enjoyable, highly, glad, durabl, pleasantly, ador, enjoy, easy
good:	passiv, transformed, directv, gifted, flight, do, educator, registration
expensive:	millhon, cheesecakes, clay, portrait, misframed, displays, unbeleiv, face
cheap:	barely, boring, hardli, unfortun, awful, return, painful, cracked
awful:	wast, died, bore, junk, defect, boring, worse, waste
return:	worst, boring, junk, disappointment, unfortunately, poorly, horrible, returning

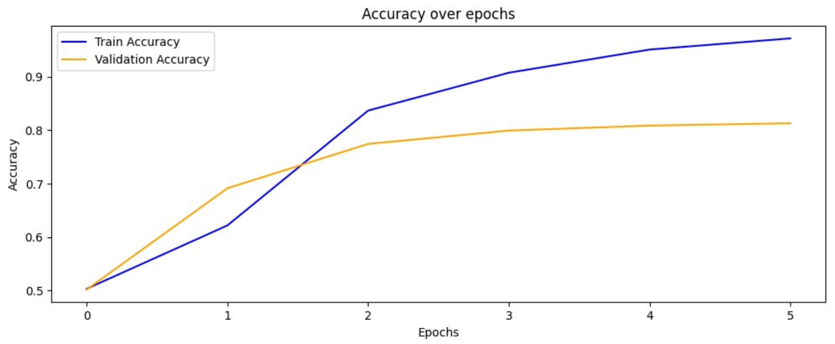
Model: Model_015

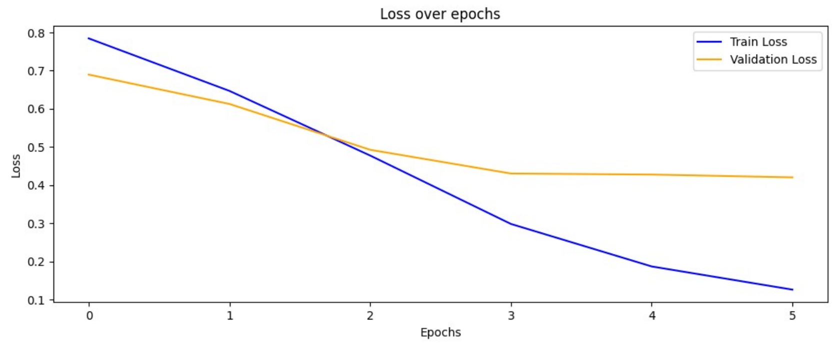
Model Size: 246.95 MB

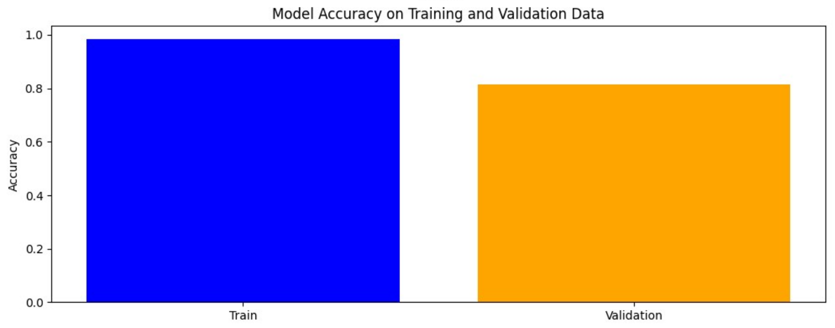
Total Parameters: 21576101

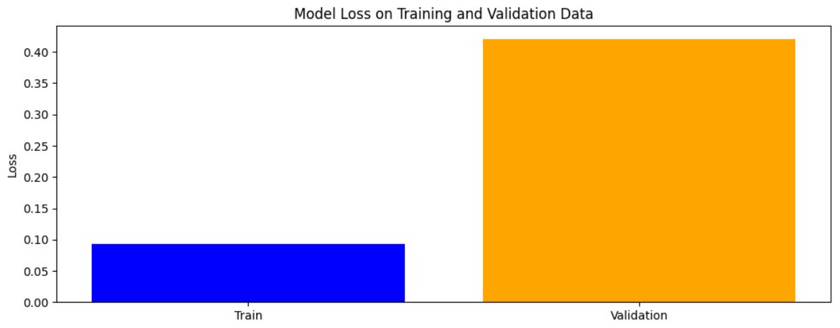
Hyperparameters

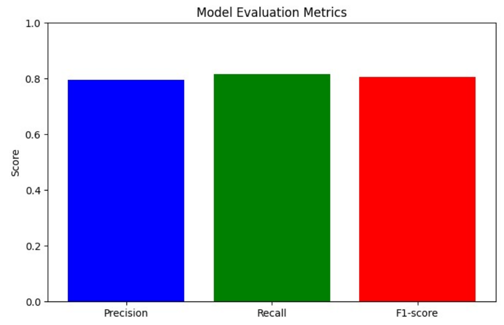
Embedding Dimension: 50
Dense (hidden) Units: 50
Dense (hidden) Activation: relu
Dense (output) Units: 1
Dense (output) Activation: sigmoid
Learning Rate: 0.0005
Test Size: 0.2
Epochs: 6
Batch Size: 128



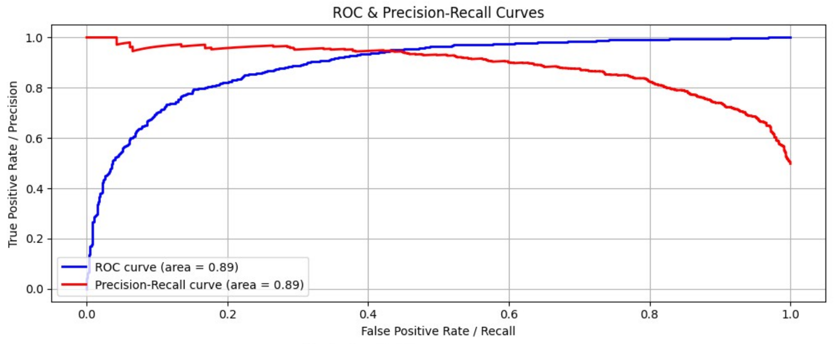


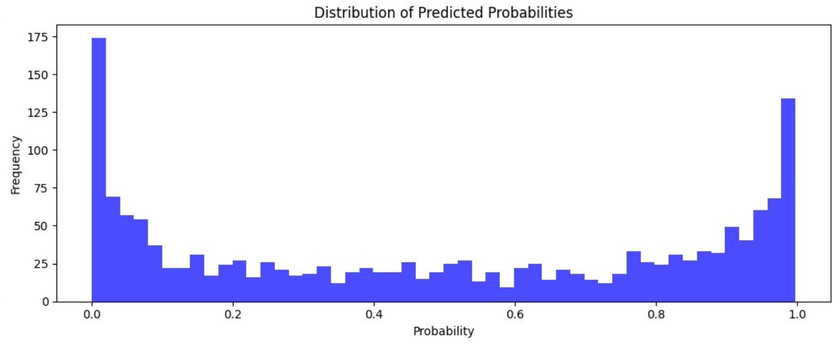


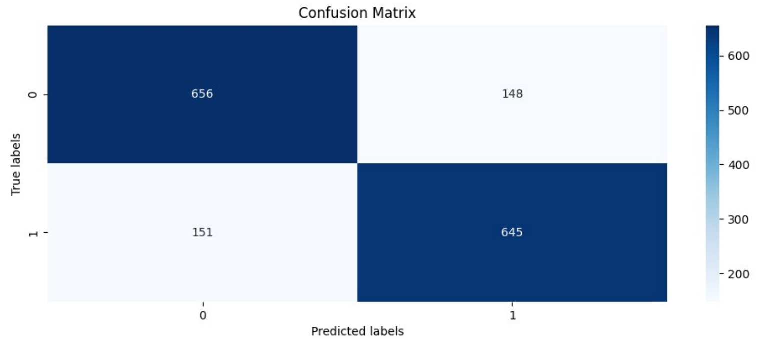




Precision: **0.7946**
Recall: **0.8166**
F1-score: **0.8055**







Similar Words Research

great:	wonderful, amaz, highly, easy, delicious, son, perfect, solid
perfect:	excellent, solid, awesome, fantast, best, amazing, glad, highli
happy:	enjoyable, highly, glad, durabl, pleasantly, ador, enjoy, easy
good:	passiv, transformed, directv, gifted, flight, do, educator, registration
expensive:	millhon, cheesecakes, clay, portrait, misframed, displays, unbeleiv, face
cheap:	barely, boring, hardli, unfortun, awful, return, painful, cracked
awful:	wast, died, bore, junk, defect, boring, worse, waste
return:	worst, boring, junk, disappointment, unfortunately, poorly, horrible, returning

Model Performance

External Validation:

The model was also tested on a set of sentences that were not included in the initial training database. This provides additional confidence in its ability to generalize well to new, unseen data.

Positive Reviews:

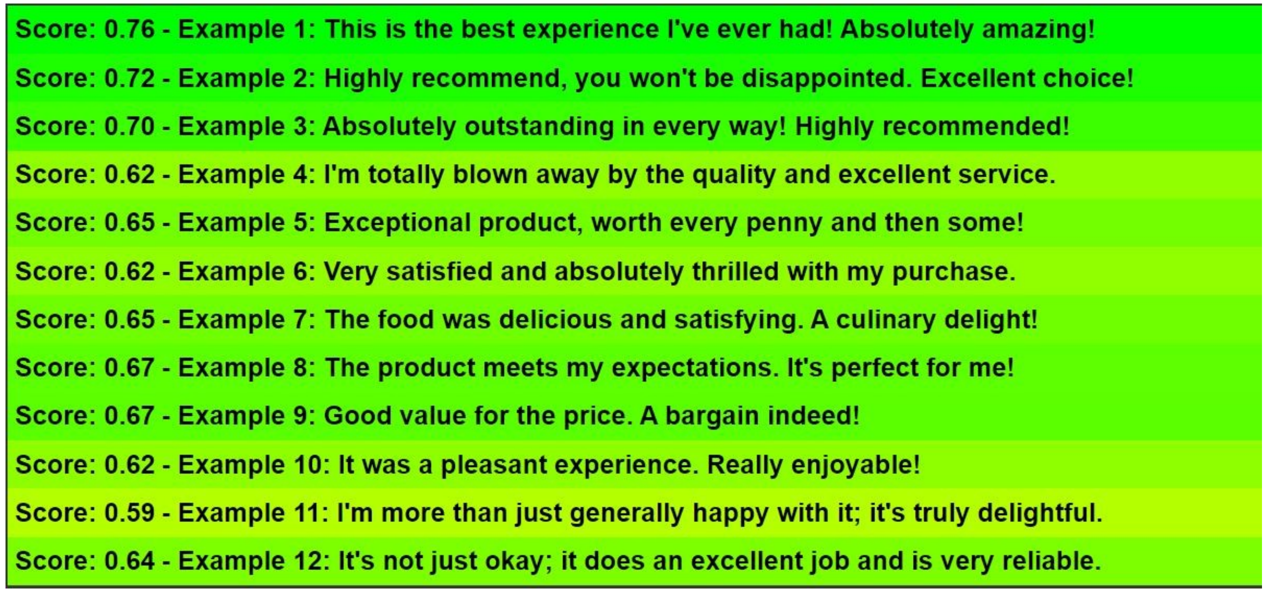
The model consistently scored positive reviews above 0.59, with a high of 0.76, indicating strong performance in identifying positive sentiments.

Negative Reviews:

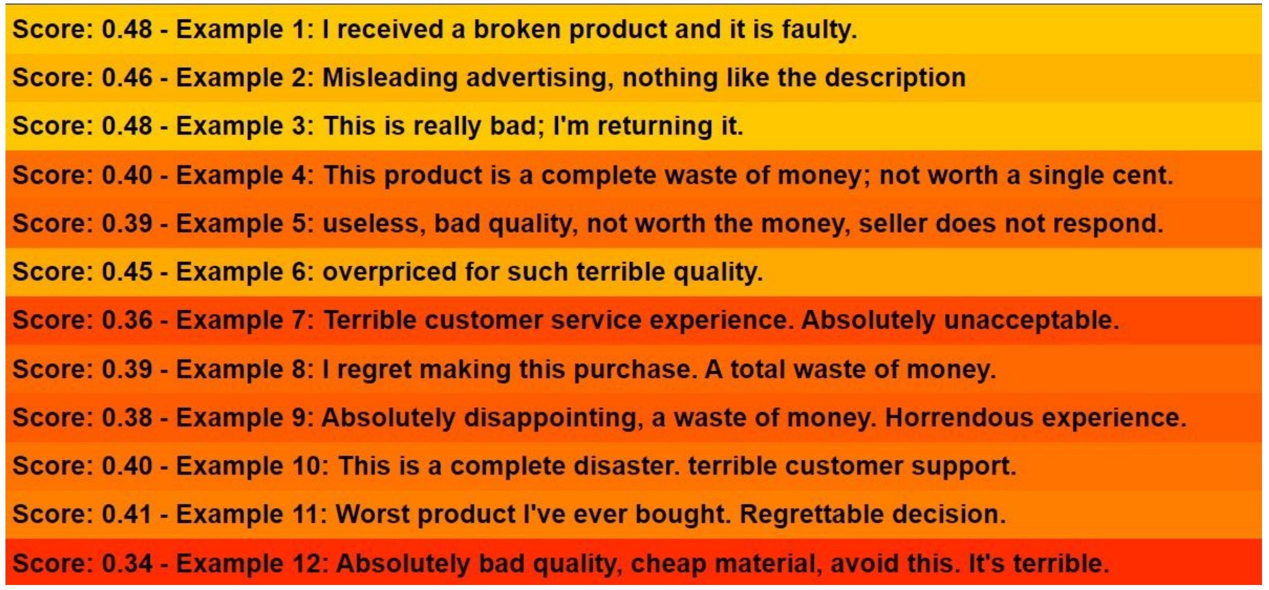
For negative reviews, the model scored them below 0.50, with the lowest being 0.34, which demonstrates its ability to categorize negative sentiments.

Overall Assessment:

The model appears to be highly effective in classifying both positive and negative reviews. It shows high confidence in identifying positive sentiments and reasonable accuracy in flagging negative sentiments. While there's always room for improvement, especially for borderline cases, the model's current performance makes it a reliable tool for practical applications in sentiment analysis.



Score: 0.76 - Example 1: This is the best experience I've ever had! Absolutely amazing!
Score: 0.72 - Example 2: Highly recommend, you won't be disappointed. Excellent choice!
Score: 0.70 - Example 3: Absolutely outstanding in every way! Highly recommended!
Score: 0.62 - Example 4: I'm totally blown away by the quality and excellent service.
Score: 0.65 - Example 5: Exceptional product, worth every penny and then some!
Score: 0.62 - Example 6: Very satisfied and absolutely thrilled with my purchase.
Score: 0.65 - Example 7: The food was delicious and satisfying. A culinary delight!
Score: 0.67 - Example 8: The product meets my expectations. It's perfect for me!
Score: 0.67 - Example 9: Good value for the price. A bargain indeed!
Score: 0.62 - Example 10: It was a pleasant experience. Really enjoyable!
Score: 0.59 - Example 11: I'm more than just generally happy with it; it's truly delightful.
Score: 0.64 - Example 12: It's not just okay; it does an excellent job and is very reliable.



Score: 0.48 - Example 1: I received a broken product and it is faulty.
Score: 0.46 - Example 2: Misleading advertising, nothing like the description
Score: 0.48 - Example 3: This is really bad; I'm returning it.
Score: 0.40 - Example 4: This product is a complete waste of money; not worth a single cent.
Score: 0.39 - Example 5: useless, bad quality, not worth the money, seller does not respond.
Score: 0.45 - Example 6: overpriced for such terrible quality.
Score: 0.36 - Example 7: Terrible customer service experience. Absolutely unacceptable.
Score: 0.39 - Example 8: I regret making this purchase. A total waste of money.
Score: 0.38 - Example 9: Absolutely disappointing, a waste of money. Horrendous experience.
Score: 0.40 - Example 10: This is a complete disaster. terrible customer support.
Score: 0.41 - Example 11: Worst product I've ever bought. Regrettable decision.
Score: 0.34 - Example 12: Absolutely bad quality, cheap material, avoid this. It's terrible.

Ethical Consideration

- Privacy and Data Consent: Obtain permission to use data, especially from sources like social media.
- Transparency and Explainability: Provide clear explanations for sentiment predictions.
- Context Sensitivity: Recognize limitations in understanding context, sarcasm, and cultural nuances.
- Misuse and Manipulation: Guard against using sentiment analysis to deceive or manipulate public opinion.
- Informed Decision-Making: Communicate limitations to prevent overreliance on analysis outcomes.
- Unintended Consequences: Monitor and adjust to prevent unintended effects, like amplifying negativity.
- Data Security: Implement strong security measures to protect sensitive sentiment data.
- Long-Term Effects: Reflect on how widespread sentiment analysis might shape communication over time.