Neural Network & Deep Learning

Sentiment Analysis

Instructor : Dr Manas Das

Deepesh Yadav

17th October, 2024

Introduction

As consumers increasingly rely on online reviews to inform their purchasing decisions, the importance of sentiment analysis has become paramount.

This project focuses on conducting sentiment analysis on reviews of ear wearable products sourced from Amazon, aiming to classify user sentiments into positive (1) and negative (0) categories.

By leveraging deep learning techniques and algorithms such as LSTM, we aim to develop an accurate model that not only classifies sentiment effectively but also offers insights into user experiences and preferences.

This report details the methodology employed in the project, including data collection, model architecture, performance evaluation, and the implementation of an accessible user interface.

Task 1: Data Collection & Data Preprocessing

1.1. Data Collection

Data Source

Note 👍:

- We did not take any dataset from any source like kaggle etc.
- We have built our own python scraper to scrape the reviews from the amazon website with browser automation using the selenium webdriver.

We have taken the ear wearables from 7 music brands such as Boat, JBL, Apple, Boult, Ubon etc.

Why only with same products (ear wearables)

- So we are doing sentiment analysis with the same type of the products because we will be using the Deep Learning model (LSTM) which will get the semantic information from the reviews of the same category of the products.
- We did not take any other product reviews because that won't align with our problem statement and the industrial use case.

Web Scraping Tools:

The scraping process utilized libraries such as ***Selenium*** and ***web driver*** requests in Python to extract data programmatically. The following information was collected:

- **Review Text:** User's written reviews about the product.
- **Review Rating:** A numerical rating (e.g., 1 to 5 stars).
- **Sentiment Label:** A binary label (0 for negative sentiment, 1 for positive sentiment) derived from the review rating.
- **Data Volume:** The scraping process yielded approximately 21,825 reviews

1.2 Data Preprocessing

Text Cleaning:

- **Lowercasing**: Convert all text to lowercase to maintain consistency.
- **Removing Punctuation**: Eliminate punctuation marks that do not contribute to sentiment.
- **Removing Emojis:** We have removed the emoji from the dataset because the number of emojis was very less, it was not frequent. So to reduce unnecessary complexity of the model by mapping the emoji with the text using the “emoji” python library.
- **Removing Stop Words**: Remove common stop words (e.g., "the", "is", "and") that do not affect sentiment meaning.
- **Lemmatization/Stemming**: Reduce words to their base or root form to handle different word variations.

Data Labeling:

- Convert review ratings into sentiment labels (e.g., ratings 1-3 as 0, ratings 4-5 as 1).

Train-Test Split: The preprocessed data is split into training (80%) and testing (20%) sets to ensure that the model is validated on unseen data.

Note: *The code is explained in the jupyter file*

Task 2: Deep Learning Model Architecture

2.1 Model Selection

For this project, we chose a Long Short-Term Memory (LSTM) model, a type of recurrent neural network (RNN) well-suited for sequential data like text. LSTMs are effective at capturing long-range dependencies, making them ideal for sentiment analysis tasks.

2.2 Model Architecture

The proposed architecture includes the following layers:

- **Embedding Layer**: This layer transforms input sequences of integers (word indices) into dense vectors of fixed size.
- **LSTM Layer**: This layer processes the sequences and learns the temporal dependencies.
- **Dropout Layer**: A dropout layer is included to reduce overfitting during training.

- **Dense Layer**: The output layer uses a sigmoid activation function for binary classification.

Layer (type)	Output Shape	Param #
embedding_12 (Embedding)	(None, 200, 128)	640,000
lstm_20 (LSTM)	(None, 128)	131,584
dense_20 (Dense)	(None, 1)	129

Total params: 771,713 (2.94 MB)

Trainable params: 771,713 (2.94 MB)

Non-trainable params: 0 (0.00 B)

The model was compiled with the **binary cross-entropy loss function** and optimized using the **Adam optimizer**. The training process used mini-batch gradient descent, with the model evaluated on a validation set to monitor performance and avoid overfitting.

2.3 Model Code

```
from keras.models import Sequential
from keras.layers import Embedding, LSTM, Dense, Dropout

# Build the model
model = Sequential()
model.add(Embedding(input_dim=5000, output_dim=128, input_length=200))
model.add(LSTM(128, dropout=0.2, recurrent_dropout=0.2))
model.add(Dense(1, activation='sigmoid'))

# Build the model with the specified input shape
model.build(input_shape=(None, 200))

# Print the model summary
model.summary()
```

Summary of Layers:

- Input ➔ Embedding ➔ LSTM ➔ Dense ➔ Output (Sigmoid)

This architecture is effective for sentiment analysis, leveraging the ability of LSTMs to capture sequential dependencies in text data.

Task 3: Model Training, Performance Evaluation

3.1 Model Training

- In this project, an LSTM model was trained on preprocessed text data to perform sentiment analysis.
- The process began by tokenizing the text data and converting it into sequences of word indices, which were then padded to ensure uniform input lengths.
- These sequences were passed through an **Embedding Layer** to map words into dense vectors that captured their semantic meaning.

```
Epoch 1/5
291/291 ━━━━━━━━━━━━━━━━━━━━ 30s 104ms/step - accuracy: 0.9235 - loss: 0.1955 - val_accuracy: 0.8797 - val_loss: 0.3001
Epoch 2/5
291/291 ━━━━━━━━━━━━━━━━━━━━ 23s 79ms/step - accuracy: 0.9341 - loss: 0.1738 - val_accuracy: 0.8866 - val_loss: 0.2997
Epoch 3/5
291/291 ━━━━━━━━━━━━━━━━━━━━ 21s 72ms/step - accuracy: 0.9362 - loss: 0.1689 - val_accuracy: 0.8905 - val_loss: 0.3067
Epoch 4/5
291/291 ━━━━━━━━━━━━━━━━━━━━ 23s 78ms/step - accuracy: 0.9446 - loss: 0.1499 - val_accuracy: 0.8961 - val_loss: 0.2985
Epoch 5/5
291/291 ━━━━━━━━━━━━━━━━━━━━ 19s 66ms/step - accuracy: 0.9481 - loss: 0.1424 - val_accuracy: 0.8933 - val_loss: 0.3120
```

3.2 Performance Evaluation

- After training, the model's performance was evaluated on a separate test set using common classification metrics such as **accuracy**, **precision**, **recall**, and **F1-score**.

```
loss, accuracy = model.evaluate(X_test, y_test)
print(f"Test Loss: {loss*100:.2f}%")
print(f"Test Accuracy: {accuracy * 100:.2f}%")
✓ 2.1s

182/182 ━━━━━━━━━━━━━━━━━━━━ 2s 11ms/step - accuracy: 0.9094 - loss: 0.2689
Test Loss: 29.08%
Test Accuracy: 90.32%
```

Task 4: Explainability and Interpretability

4.1 Why LSTM over ANN ?

- **Memory Capability**: LSTMs have memory cells that store information over time, making them suitable for tasks involving sequences, like time series forecasting and NLP.
- **Temporal Dependency Learning**: LSTMs can learn and retain long-term dependencies in sequential data, unlike ANNs which treat each input independently.
- **Vanishing Gradient Solution**: LSTM's architecture mitigates the vanishing gradient problem, which is common in traditional RNNs and ANNs, allowing them to learn longer sequences effectively.
- **Gate Mechanisms**: LSTMs use input, output, and forget gates to control the flow of information, helping to remember relevant information and forget irrelevant data over time.
- **Better Performance on Time Series**: For tasks like stock market prediction or language modeling, LSTMs outperform ANNs because they leverage the time relationships between data points.

4.2 Explainability

While LSTMs provide a powerful tool for sentiment analysis, their complexity makes them less interpretable compared to simpler models. However, we can still derive some level of explainability through the following techniques:

Gates Analysis:

- LSTMs consist of input, forget, and output gates that determine how much information is retained or discarded at each time step. By analyzing the activation of these gates during training, we can gain insights into what information the model considers important for its predictions.
- For instance, if the forget gate shows high activation for certain past inputs, it indicates that the model is intentionally discarding this information, which can help understand how the model behaves with respect to specific sequences.

Attention Mechanisms:

- Though traditional LSTMs do not inherently include attention mechanisms, they can be integrated.
- Attention allows the model to weigh the importance of different parts of the input sequence when making predictions.
- By visualizing attention weights, we can gain insights into which words or phrases are influencing the model's decisions.

Layer Activation Visualization:

- Analyzing the activation of different layers within the LSTM can help understand how the model processes input data.
- For example, we can visualize activations after certain LSTM layers to see how they respond to different words in a review, offering clues about which parts of the input are driving predictions.

4.3 Loss of the Interpretability in LSTM model:

Despite the explainability techniques mentioned, LSTMs face significant challenges in interpretability due to their architecture:

Complexity of Operations:

- LSTMs involve complex mathematical operations and multiple layers of non-linear transformations.
- This complexity makes it difficult to trace back the exact contribution of individual input features (e.g., words) to the final output.
- Unlike simpler models where weights can be directly interpreted, LSTM's internal state is not easily understood in terms of the input features.

Feature Interaction:

- In LSTMs, features (words) interact in a way that is not straightforward.
- A word may contribute to the sentiment through interactions with preceding words.
- The hidden states of the LSTM evolve over time based on the entire sequence, making it challenging to isolate the impact of specific words or phrases on the final sentiment classification.

Temporal Dependencies:

- The temporal dependencies that LSTMs capture can obscure interpretability. The sentiment of a particular word may depend on its context within the sentence or paragraph, complicating efforts to explain how each feature contributes to the final sentiment score.

Task 5: User Interface Development, Deployment and Accessibility, Continuous Improvement, Ethical Considerations

5.1 User Interface Development

Creating an intuitive and user-friendly interface for users to input product reviews and receive sentiment analysis results such as type of review and the sentiment score with Emoji.

Front-End Framework:

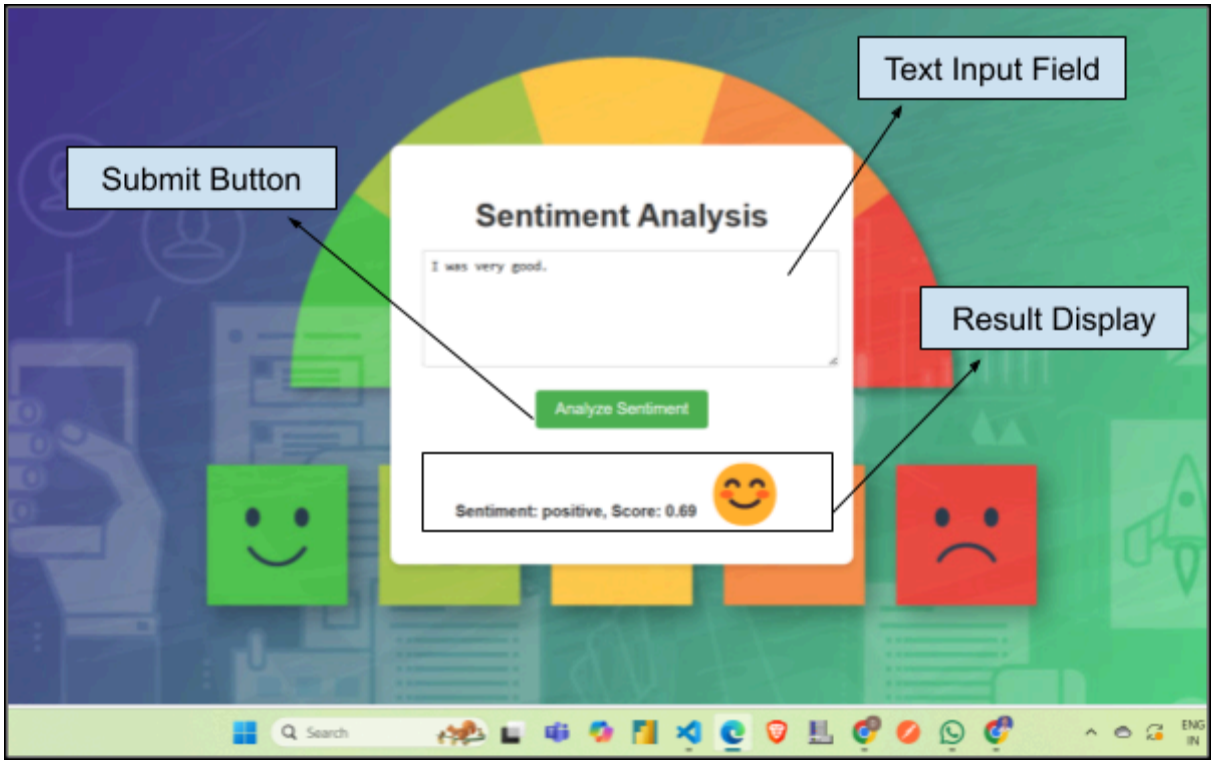
- We used **HTML, CSS, and JavaScript** for building the front-end.
- These technologies help create a responsive, interactive, interface for better user experience..

Backend Framework:

- The model is very simple so we have used **Flask** for serving this model and handling user inputs.
- Flask is lightweight and well-suited for smaller projects, making it a good choice for rapid development.

Interface Features:

- **Input Field**: A text box where users can input their review text.
- **Submit Button**: Users can submit their reviews for sentiment analysis.
- **Result Display**: Display the predicted sentiment as "Positive" or "Negative" along with a confidence score (e.g., 85% Positive).



5.2 Deployment and Accessibility

- For the Deployment we didn't deploy on the cloud rather we built the flask model on local system.

Setting up the project file structure 👍

```
Flask Project/
|
├── app.py                          # Main Flask application file
├── requirements.txt                # Dependencies for your application
├── templates/                      # Folder for HTML templates
|   ├── index.html                   # Input form for user reviews
├── sentiment_analysis_model_main.h5  # Your trained model
└── main_tokenizer.pickle           # Your tokenizer
```

- We trained the model in the google Colab then we saved that model as “sentiment_analysis_model_main.h5”.
- We also saved the tokenizer file as “main_tokenizer.pickle”

The UI is designed with keeping the accessibility in mind. Key considerations include keyboard navigation support and clear text areas for user input.

5.3 Continuous Improvement

- As we know that the LSTM is a data hunger algorithm, which needs a lot of data to train the model for better accuracy.
- We can scrape the data from different sites too and increase the variety of the products.
- We can also improve the model with different hyperparameters values to generalize the model for the future data.
- The interface can also be optimized by gathering user feedback and incorporating analytics tools to track behavior.
- We can also deploy the model on the clouds like AWS, Heroku etc.

5.4 Ethical Considerations

- In this project we are scraping the data from very popular websites like Amazon.com but other smaller website's servers do not allow the user to scrape the data for example https://stufin.in/
- So we need to keep in mind while scraping the data from websites it might contain some malicious data.
- We should read the policy of the website and keep in mind that we are not scraping the sensitive data of the user.
- Ethically, the project should prioritize user privacy by ensuring that no sensitive data is stored or misused.
- Additionally, efforts should be made to reduce bias in the sentiment model by continually evaluating it for fairness across different types of reviews.
- Transparency is key; users should be informed about the limitations of the AI model and how their data is used.

Conclusion

The sentiment analysis project on ear wearable product reviews effectively demonstrates the use of deep learning, specifically LSTM models, to understand customer feedback.

By classifying reviews as positive or negative, the model offers valuable insights for improving product quality.

Starting from data collection and preprocessing to training and deploying the LSTM model, the project achieved a balanced and accurate sentiment analysis system.

Local deployment allows real-time testing, while accessibility measures ensure inclusivity for all users.

Future improvements could focus on fine-tuning the model, adding interpretability features, and scaling the application.

Overall, this project highlights the potential of deep learning in transforming user reviews for the product into actionable insights which can be used by the company.