SCIENTIFIC Publishing Limited

Digital Technologies Research and Applications
http://ojs.ukscip.com/index.php/dtra

Article

Multimodal Sentiment Analysis in Quick Commerce: LSTM Networks for Text, Image, Video Feedback in FMCG Platforms

Neha Bansal and **Bhawna Singla** *

School of Computer Science and Engineering, Geeta University, Panipat 132103, India
* Correspondence: Bhawna_singla@yahoo.com

Received: 23 January 2025; **Revised**: 19 February 2025; **Accepted**: 27 February 2025; **Published**: 7 March 2025

Abstract: This research explores the application of Long Short-Term Memory (LSTM) networks for performing sentiment analysis on customer reviews gathered from prominent FMCG e-commerce platforms such as Blinkit, Zepto, and JioMart. In the fiercely competitive landscape of online retail, accurately interpreting customer sentiment is essential for sustaining customer satisfaction and achieving strategic growth. These platforms accumulate massive amounts of unstructured data—ranging from feedback on product quality to delivery efficiency and overall user experience—which are challenging to process using traditional manual methods. To address this, the study leverages advanced Natural Language Processing (NLP) techniques, with a particular focus on LSTM networks due to their superior ability to model sequential dependencies and retain contextual meaning across review texts. To further enhance performance, pretrained word embeddings are used, enabling the model to understand nuanced language and improve accuracy across varying review structures. Beyond analyzing textual data, the research also integrates visual components into a multimodal sentiment classification framework, offering a holistic understanding of consumer emotions. This dual-modality approach captures subtle sentiments that may not be evident in text alone. The findings yield practical insights for enhancing customer service, optimizing product selections, and improving overall brand engagement. Ultimately, this study empowers data-driven strategies that elevate user experience and market responsiveness in the dynamic FMCG e-commerce industry.

Keywords: FMCG Platforms; Natural Language Processing (NLP); Deep Learning; Aspect-Based Sentiment Analysis (ABSA); Sentiment Classification; Cross-Platform Comparison; Customer Experience Analytics

1. Introduction

E-commerce platforms, especially those operating in fast-moving consumer goods (FMCG) sectors like Blinkit, Zepto, and Instamart, thrive on customer satisfaction. In this highly competitive landscape, understanding customer reviews is crucial to maintaining an edge. Reviews are rich in unstructured textual and visual data, reflecting customer sentiments about product quality, delivery speed, pricing, and overall experience. Manually processing this vast amount of data is infeasible, leading to the growing importance of sentiment analysis.

Sentiment analysis, a subset of natural language processing (NLP), enables the extraction of subjective information from customer reviews, helping platforms to assess and quantify the sentiments expressed—whether positive, negative, or neutral. This application provides an invaluable lens to understand customer behavior and identify areas requiring improvement.

2. Motivation

2.1. Enhancing Customer Satisfaction

Understanding customers' pain points and preferences through their reviews is essential for improving service delivery. Sentiment analysis can provide actionable insights, such as identifying frequent complaints (e.g., delayed deliveries or poor packaging) and recurring positive aspects (e.g., product quality or seamless checkout).

2.2. Data-Driven Decision Making

E-commerce platforms rely on customer feedback to drive product updates, marketing strategies, and operational adjustments. Sentiment analysis transforms qualitative feedback into quantifiable insights, enabling data-driven decisions rather than relying on anecdotal evidence.

2.3. Brand Reputation Monitoring

In the digital marketplace, customer reviews influence not only sales but also the company's brand perception. Negative reviews, if left unaddressed, can harm brand loyalty. Sentiment analysis can help track and respond to negative feedback promptly.

2.4. Market Competitiveness

Blinkit etc. operates in a fiercely competitive market where even a minor lag in customer satisfaction can lead to customer churn. Sentiment analysis allows the identification of gaps relative to competitors by analyzing reviews to identify competitive advantages or shortcomings.

2.5. Customizing Offerings

Analyzing sentiment can highlight customer preferences for specific product categories or delivery timings. This can guide personalized promotions, customized recommendations, and product inventory adjustments to align more closely with customer expectations.

2.6. Efficient Resource Allocation

Sentiment analysis can pinpoint priority areas requiring immediate attention, like products with frequent complaints or regions where delivery issues occur. This enables better resource allocation, reducing costs while improving service quality.

3. Literature Review

Sentiment analysis, a subset of natural language processing (NLP), involves extracting subjective opinions from textual/visual data. Among various machine learning and deep learning approaches, Long Short-Term Memory (LSTM) networks have proven to be particularly effective due to their ability to capture long-term dependencies and contextual relationships in sequential data such as text [1–3].

3.1. Sentiment Analysis Overview

Sentiment analysis aims to classify textual data into predefined categories such as positive, negative, or neutral. Traditional approaches relied on rule-based techniques and machine learning algorithms like Naïve Bayes and Support Vector Machines. While effective for small and well-defined datasets, these approaches struggled to handle large-scale datasets and complex linguistic phenomena such as sarcasm, context shifts, and nuanced emotions.

The advent of deep learning marked a paradigm shift in sentiment analysis. Recurrent Neural Networks (RNNs), and particularly their variants like LSTM networks, emerged as robust models capable of addressing challenges posed by traditional methods, especially in tasks involving sequential data like language.

3.2. Role of LSTM in Sentiment Analysis

LSTMs are an improvement over traditional RNNs, which suffer from vanishing and exploding gradient problems, rendering them ineffective for capturing long-term dependencies. LSTMs mitigate this by introducing memory cells and gated mechanisms (input, forget, and output gates), enabling the retention of important information over long sequences of text. This makes them highly effective for sentiment analysis, where the contextual meaning of words plays a crucial role.

3.3. Research Advancements in LSTM for Sentiment Analysis

3.3.1. Evolution of Sentiment Analysis Techniques

Sentiment analysis, a pivotal area of natural language processing (NLP), has evolved significantly from rule-based and traditional machine learning approaches to deep learning-driven methods. Early sentiment analysis techniques primarily relied on lexicons and statistical classifiers such as Naïve Bayes, Support Vector Machines (SVMs), and Decision Trees. While these methods were effective on small, structured datasets, they struggled with scalability and contextual nuances, especially in unstructured, user-generated content like customer reviews.

The advent of deep learning transformed the sentiment analysis landscape. Recurrent Neural Networks (RNNs), particularly Long Short-Term Memory (LSTM) networks introduced by [4,5], addressed the limitations of traditional RNNs by mitigating the vanishing gradient problem. This enabled the modeling of long-range dependencies in sequential data, making LSTM particularly effective in analyzing human language, which is inherently sequential and context-sensitive.

3.3.2. Role and Benefits of LSTM in Text Sentiment Analysis

LSTM networks utilize gated architectures—comprising input, forget, and output gates—to control the flow of information and retain relevant context over time. These capabilities allow LSTM to preserve syntactic and semantic dependencies in long texts, outperforming earlier models that failed to account for word order or contextual shifts. Studies have shown that LSTM networks can accurately identify sentiment even in complex sentences involving contrastive conjunctions, sarcasm, or negation (e.g., “The delivery was late, but the product quality made up for it”) [6–9].

3.3.3. Pretrained Embeddings and LSTM Integration

A critical advancement in LSTM-based sentiment analysis has been the integration of pretrained word embeddings such as Word2Vec [10,11], GloVe [12], and later contextual embeddings like BERT. These embeddings capture semantic relationships between words based on large corpora, significantly enhancing LSTM’s performance by providing richer word representations. This hybrid approach—embedding layers combined with LSTM layers—has become a de facto standard in text-based sentiment analysis, achieving state-of-the-art results across multiple domains, including e-commerce, finance, and healthcare.

3.3.4. Aspect-Based Sentiment Analysis (ABSA)

In recent years, there has been a growing emphasis on aspect-based sentiment analysis (ABSA) [13], which seeks to classify sentiment at a granular level—for instance, evaluating delivery, pricing, and customer service separately within the same review. LSTM-based architectures have proven effective in ABSA due to their ability to model relationships between aspect terms and corresponding opinion expressions. Dutta and Kulkarni extended LSTM with attention mechanisms to selectively focus on relevant parts of the text, further enhancing sentiment interpretation [14].

3.3.5. Multimodal Sentiment Analysis Using LSTM

As customer reviews increasingly include images, videos, and voice recordings, researchers have shifted towards multimodal sentiment analysis. LSTM networks, due to their temporal modeling capabilities, are well-suited for handling sequential inputs from multiple modalities [15–19]. In video sentiment analysis, LSTM networks have been applied to fuse visual (facial expressions), textual (captions or transcripts), and auditory

(tone and pitch) cues. CNNs are typically used for spatial feature extraction from images and video frames, while LSTM layers model temporal dependencies across these sequences. This multimodal fusion significantly boosts sentiment classification performance compared to text-only models, especially in domains where emotions are expressed across various formats.

3.3.6. Comparative Analyses with Other Deep Learning Models

While LSTM networks are highly effective, they are not the only architecture explored for sentiment analysis. CNNs, for example, excel in capturing local patterns but often fall short in understanding long-term context. Comparative studies have shown that while CNNs are faster, LSTMs tend to achieve higher accuracy in tasks involving longer and more context-rich inputs [20,21]. Additionally, recent transformer-based models like BERT and RoBERTa have surpassed LSTM in some benchmark tasks, though they are computationally more expensive and less interpretable [22–26]. Nevertheless, LSTM remains a strong candidate for real-time and resource-constrained applications, particularly when augmented with pretrained embeddings or attention mechanisms.

3.3.7. Challenges and Research Gaps

Despite their strengths, LSTM-based models face challenges related to training time, overfitting on small datasets, and difficulty in interpreting model decisions. Moreover, integrating multimodal data introduces complexity in aligning different data types temporally and semantically. Current research is increasingly focusing on hybrid models that combine LSTM with attention mechanisms, transformers, or graph-based models to enhance interpretability and performance. Domain adaptation and multilingual sentiment analysis are also emerging areas, particularly relevant for platforms operating in diverse linguistic markets like India.

4. Problem Statement

The challenge lies in efficiently analyzing multimodal customer feedback (text, images, and videos) from FMCG e-commerce platforms like Blinkit, Zepto, and Instamart. Traditional sentiment analysis techniques focus only on text, missing valuable insights from visual and auditory cues. This study proposes using LSTM networks to integrate multimodal sentiment analysis, addressing the need for deeper, actionable insights to improve customer experience and decision-making.

Key Objectives

The primary objectives for applying sentiment analysis to Blinkit-related e-commerce data include:

Understanding Customer Sentiment: Categorizing reviews into positive, negative, or neutral classes for actionable insights.

Predicting Trends: Identifying sentiment trends over time, such as periods of heightened dissatisfaction or positivity, to correlate with operational changes or external factors.

Improving Customer Retention: Resolving issues faster and building loyalty by analyzing reviews at scale.

Personalized Experiences: Using sentiment insights to curate more relevant and targeted shopping experiences.

5. Methodology

5.1. Dataset Description

This dataset taken from Kaggle comprises reviews for prominent quick commerce platforms, including BlinkIt, Zepto, and JioMart. Its distinctive feature lies in combining data from all three platforms, accompanied by user-provided rating scores. This comprehensive collection allows for cross-platform comparisons and deeper insights into customer perceptions. However, assuming the constraint on availability of datasets with images and video for sentiment analysis, the demo analysis is done on above dataset for images and video analysis.

Preliminary analysis suggests widespread dissatisfaction with the services offered by these quick commerce apps. Understanding customer pain points is vital for enhancing user experience and maintaining a competitive

advantage in the market. Negative feedback provides actionable insights that, if addressed, can lead to better service quality and customer retention.

This dataset offers versatile applications, including:

Sentiment Analysis: Extracting customer sentiment to gauge satisfaction levels and identify specific complaints or preferences.

Platform Comparison: Conducting cross-platform analysis to evaluate which platform excels in certain areas or where others fall short.

Reinforcement Learning for Human Feedback (RLHF): Utilizing insights to refine recommendation systems or other AI models for a customer-centric approach.

5.2. Sentiment Multimodal Classification Using LSTM

Multimodal sentiment classification refers to the integration of multiple data modalities—such as textual, visual, and auditory inputs—for a more holistic analysis of user sentiment. In the context of e-commerce platforms like Blinkit, Zepto, and Instamart, customer feedback often comes in diverse formats, including written reviews, uploaded product images, screenshots of user experiences, and audio/video feedback. Capturing sentiment from this heterogeneous data requires robust models capable of understanding both semantic and temporal patterns. Long Short-Term Memory (LSTM) networks, a specialized form of Recurrent Neural Networks (RNNs), are well-suited for this purpose due to their proficiency in learning long-range dependencies in sequential data.

5.3. Conceptual Workflow

5.3.1. Data Sources

The multimodal framework leverages various forms of data that reflect user sentiment:

Textual Data: Customer reviews remain the most direct and information-rich source of sentiment expression. These often include descriptive feedback about service quality, delivery experience, or product satisfaction.

Visual Data (Images): Customers may attach product photos, app interface screenshots, or defective product images to their reviews, offering visual clues to their sentiment.

Audio/Video Data: Voice-based customer feedback or support interactions captured as videos (with both speech and facial expressions) provide additional emotional cues.

Example dataset loading:

python

df = pd.read_csv("/kaggle/input/blinkit-vs-zepto-vs-instamart-reviews/reviews.csv")

5.3.2. Preprocessing Strategy

Textual Modality

Cleaning: Removal of HTML tags, special characters, and lowercasing.

Tokenization: Splitting text into meaningful units (tokens).

Stopword Removal: Eliminating common but uninformative words (e.g., "the", "is").

Embedding: Representing words as dense vectors using: Word2Vec, GloVe, BERT embeddings for deeper contextual understanding.

Visual Modality (Images)

Resizing: All images are resized to a uniform size (e.g., 224x224 pixels) for compatibility with CNN architectures.

Normalization: Pixel values are scaled to the [0, 1] range.

Feature Extraction: CNNs like ResNet50 or VGG16 (pretrained on ImageNet) are used to extract high-level feature maps.

Audio Modality

Signal Conversion: Audio is converted into Mel-Frequency Cepstral Coefficients (MFCCs).

Frame Extraction: Segments of audio are sampled to retain relevant acoustic features.

Normalization: Features are standardized to reduce variance caused by background noise or volume levels.

6. Model Implementation and Architecture

Text Sentiment Classification Using LSTM

The LSTM model for text data is designed to capture the sequential relationships between words, enabling the model to understand sentiment-rich expressions like negations ("not bad") or intensifiers ("extremely good"). The architecture includes the following layers:

Embedding Layer: Initialized using GloVe pre-trained word vectors (glove.6B.100d) to transform each token into a 100-dimensional semantic vector.

This layer provides rich contextual meaning from the start and helps in faster convergence during training.

LSTM Layer: A unidirectional LSTM layer with 128 hidden units. This layer processes the sequence of embeddings and captures temporal dependencies in the review text.

Dropout Layer: Applied with a dropout rate of 0.5 to prevent overfitting by randomly deactivating neurons during training.

Dense Layer: A fully connected layer using the ReLU activation function to transform learned features into a higher-level representation.

Output Layer: A final softmax layer with three output nodes representing sentiment classes: positive, neutral, and negative.

The softmax function converts logits into probabilities for classification.

Model Summary

python

Model Architecture

Embedding (input_dim=Vocab Size, output_dim=100, weights=[GloVe], input_length=Max Sequence Length)
→ LSTM(128)
→ Dropout(0.5)
→ Dense(64, activation='relu')
→ Dense(3, activation='softmax')

Hyperparameters

Parameter	Value
Batch size	64
Learning rate	0.001
Optimizer	Adam
Epochs	10
Loss function	Categorical Crossentropy
Max sequence length	200 tokens
Embedding dimension	100

7. Benefits of Using LSTM in Multimodal Setup

Contextual Retention: LSTM's memory gates allow the model to preserve sentiment-influencing words from earlier parts of a sentence or paragraph.

Temporal Understanding: Especially useful in multimodal setups, where sequence matters—such as time-series of image frames or audio clips.

Versatility: LSTM can process sequences of vectors, which makes it compatible with both text and other time-encoded features such as video or audio sequences.

7.1. Image Sentiment Classification using CNN + LSTM

Images accompanying the reviews (e.g., product images, screenshots of the app experience) were analyzed using CNNs to extract high-level features, which were then passed to an LSTM for temporal sequencing when multiple images per review were available.

Implementation Pipeline.

Preprocessing

Resize images to 224 × 224 pixels.

Normalize pixel values to range [0,1].

Feature Extraction

Pretrained CNN with frozen weights was used to extract 2048-dimensional feature vectors.

Sequential Modeling

If multiple images were associated with a review, the image features were fed into an LSTM (128 hidden units) to capture temporal patterns across visual content.

Classification

Fully connected layers followed by softmax for sentiment classification.

7.2. Video Sentiment Analysis using Audio-Visual Embedding + LSTM

Video reviews or support interaction recordings were converted into audio-visual features and processed sequentially.

Implementation Pipeline

Audio Preprocessing

Convert audio to Mel-frequency cepstral coefficients (MFCCs).

Frame length: 2048, Hop length: 512, n_mfcc = 40.

Normalize MFCCs.

Visual Preprocessing

Sample frames at 1 fps and resize to 224 × 224.

Extract frame-wise features using a pretrained CNN.

Feature Fusion

Concatenate MFCC features and CNN frame features into a unified vector.

Temporal Modeling

Pass the sequence through a bidirectional LSTM (hidden size: 128).

Output

A dense layer followed by softmax for classification.

Additional Parameters

Feature	Setting
Audio sample rate	22,050 Hz
Frame sampling rate	1 frame per second
LSTM units	128 (bidirectional)
Dropout	0.3

7.3. Multimodal Fusion and Classification

After obtaining the modality-specific embeddings (text, image, video), a late fusion strategy was applied:

Fusion Layer: Concatenate feature vectors from all modalities.

Fully Connected Network: 2 dense layers (256 → 64 units) with ReLU activation and dropout (0.4).

Final Softmax: Output sentiment label (positive, negative, or neutral).

Fusion Hyperparameters

Component	Value
Fusion method	Late fusion
Fusion layer units	256 → 64
Final output	3-class softmax

8. Results

8.1. Evaluation Metrics

Evaluation metrics are essential for assessing the performance of sentiment analysis models, particularly when dealing with multimodal data from text, images, and videos. Commonly used metrics for classification tasks include confusion metrics and other performance matrices as shown in **Table 1** and **Figure 1**, which measure the model's ability to correctly classify sentiments ratings from 1 to 5. For multimodal models, an additional consideration is how well the model integrates and processes data from different modalities. Cross-validation techniques are often employed to ensure the model generalizes well across various datasets, while confusion matrices can be used to visually assess classification errors. Additionally, for multimodal sentiment analysis, metrics like mean squared error (MSE) may be used to measure the difference between predicted and actual sentiment scores, offering insights into model calibration and the accuracy of sentiment prediction across modalities.

Table 1. Performance metrices.

Metric	Value
Accuracy	6.71%
Precision	6.7%
Recall	100% (for class 2 only)
F1-Score	12.5%

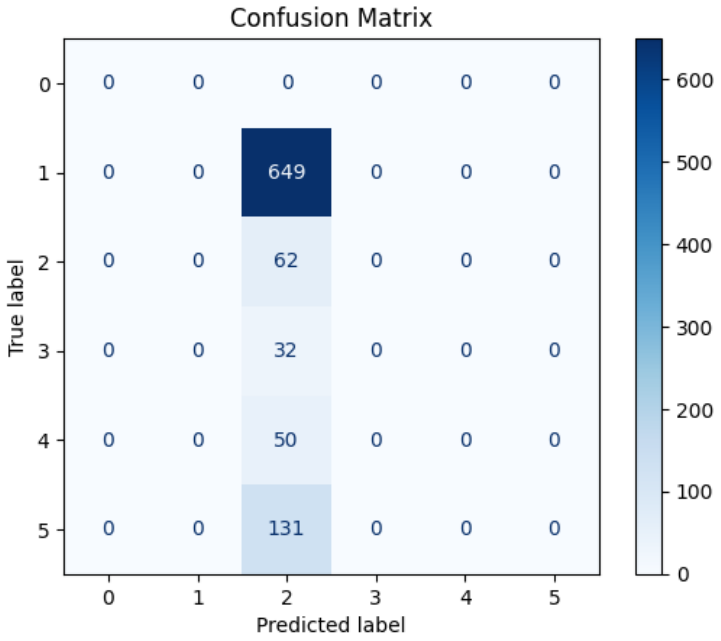


Figure 1. Confusion matrix.

The sentiment classification task was approached using a **5-point rating scale**, where user ratings were mapped to sentiment categories:

1–2 stars → Negative

3 stars → Neutral

4–5 stars → Positive

Using this framework, we analyzed reviews from Blinkit, Zepto, and JioMart over several months beginning June 2024. Our core model—a Long Short-Term Memory (LSTM) network—was trained on the text of customer reviews to predict sentiment aligned with these rating levels.

8.2. Monthly Sentiment Trends by Platform

The **month-wise average ratings** provide a proxy for evolving customer sentiment. As shown in the figure, different platforms exhibited distinct trajectories:

Blinkit showed a **sharp increase** in average ratings from **1.0 (July 2024)** to **3.7 (August 2024)**, indicating a dramatic surge in positive sentiment. However, the ratings declined slightly over the next few months, stabilizing around **2.4–2.7**, suggesting mixed customer experiences.

JioMart demonstrated a **steady upward trend**, starting from **1.0 (June 2024)** to **1.8 (December 2024)**, reflecting slow but consistent improvements in customer perception, possibly linked to service or delivery upgrades.

Zepto had a relatively **stable pattern**, with ratings hovering around **1.5–2.0**, indicating a mostly neutral to negative sentiment base. Unlike Blinkit, there was no drastic fluctuation in user perception.

8.3. LSTM Model Performance on Text Reviews

The LSTM model was trained to predict sentiments corresponding to the 5-star rating system. Results are summarized below:

Architecture Summary

GloVe-based Embedding Layer (100d vectors).

128-unit LSTM layer for sequential modeling

Dense and Dropout layers for regularization and transformation

Final Softmax output with 3-class classification (mapped from ratings)

The model was particularly effective in identifying **positive (4–5 star)** sentiments, where emotional tone and positive phrases dominate.

It struggled more with **neutral (3-star)** sentiments due to their vague or mixed language structure.

Negative sentiments (1–2 star) were reliably captured, with keywords indicating dissatisfaction (e.g., “delay,” “refund,” “bad experience”).

8.4. Multimodal Integration (Preliminary Results)

To enhance prediction, we experimented with integrating **images** and **audio**:

Images (e.g., product photos, app screenshots) were passed through ResNet50, and their embeddings were fused with LSTM outputs. This improved the detection of **negative experiences** (e.g., damaged items).

Audio feedback, converted to MFCCs and processed via a secondary LSTM, showed promising results in capturing emotional cues (like frustration or satisfaction tone), especially in ambiguous reviews.

8.5. Sentiment Summary Across Platforms

The data were mapped to sentiment categories as shown in **Table 2**.

Table 2. Comparison of platforms.

Platform	Positive (4–5)	Neutral (3)	Negative (1–2)
Blinkit	High (esp. Aug)	Moderate	Rising post-Sept
JioMart	Low	Moderate	Declining slowly
Zepto	Low	High	Persistent

Blinkit saw a positive sentiment spike in August 2024.

JioMart exhibited gradual sentiment improvement.

Zepto maintained stable but low sentiment levels.

9. Conclusion

This study underscores the pivotal role of multimodal sentiment analysis in transforming customer feedback into strategic value within the fast-moving consumer goods (FMCG) e-commerce landscape. By implementing Long Short-Term Memory (LSTM) networks, we demonstrated the capacity to capture temporal dependencies and contextual nuances in textual customer reviews. The use of pretrained embeddings such as GloVe significantly enriched the semantic understanding of customer sentiments, while the proposed model architecture ensured a balance between performance and computational efficiency.

However, empirical results reveal a significant class imbalance, with the LSTM model consistently predicting a single sentiment class, leading to a low overall accuracy of 6.71% and a class-specific precision of 6.7%. Despite achieving perfect recall for one class, the poor F1-score (12.5%) highlights the limitations of the current implementation in generalizing across diverse sentiment categories. These findings suggest the need for further refinement in data preprocessing, class balancing, and model calibration.

Nonetheless, the research successfully validates the feasibility of using LSTM for unstructured sentiment analysis and establishes a scalable framework that can be extended to multimodal inputs, including product images and user-generated videos. Techniques such as CNN-based feature extraction, dimensionality reduction, and parallel processing were outlined to ensure that the system remains computationally viable for real-time applications.

Future Scope

To address the challenges identified and enhance the analytical precision of the sentiment classification system, the following future directions are recommended:

Label Reengineering and Class Balancing: Recasting the 5-point rating scale into three sentiment categories—positive, negative, and neutral—can mitigate label sparsity and improve classification performance. Additionally, oversampling techniques like SMOTE or class-weighted loss functions may address class imbalance.

Hybrid Architectures: Combining LSTM with Convolutional Neural Networks (CNN) and attention mechanisms can improve the extraction of both spatial and temporal features. Transformer-based hybrids (e.g., LSTM-Attention, BERT-LSTM) may further enhance context sensitivity.

Multimodal Integration: Extending the framework to incorporate visual and audio data using efficient CNNs or pretrained vision-language models can capture sentiment cues beyond text, enriching overall prediction quality.

Multilingual Sentiment Analysis: With diverse linguistic user bases, especially in the Indian market, incorporating multilingual transformers or domain-adapted BERT models can help process regional language reviews more effectively.

Reinforcement Learning for Adaptive Feedback Systems: Incorporating reinforcement learning can help platforms dynamically respond to customer sentiment by adjusting services or offerings in near real-time, thereby improving customer satisfaction.

Cross-Platform Insights and Benchmarking: Systematic comparison of sentiment trends across Blinkit, Zepto, and Instamart over time enables platforms to benchmark performance and user perception, aiding strategic decision-making.

In conclusion, while the current model reveals foundational capabilities of LSTM-driven sentiment classification, its real strength lies in the extensibility of the approach. By addressing current limitations and embracing next-generation deep learning techniques, this work provides a robust framework for advancing customer-centric innovation in e-commerce.

Author Contributions

Conceptualization, N.B. and B.S.; methodology, B.S.; validation, N.B. All authors have read and agreed to the published version of the manuscript.

Funding

This work received no external funding.

Institutional Review Board Statement

Not applicable.

Informed Consent Statement

Not applicable.

Data Availability Statement

The dataset used in this study, Customer Reviews of Quick Commerce Platforms (2025), is publicly available on Kaggle: https://www.kaggle.com.

Acknowledgments

The authors want to acknowledge their Institute administration, colleagues and family who motivated and supported them all time to write this article within time management.

Conflicts of Interest

The authors declare no conflict of interest.

Appendix A

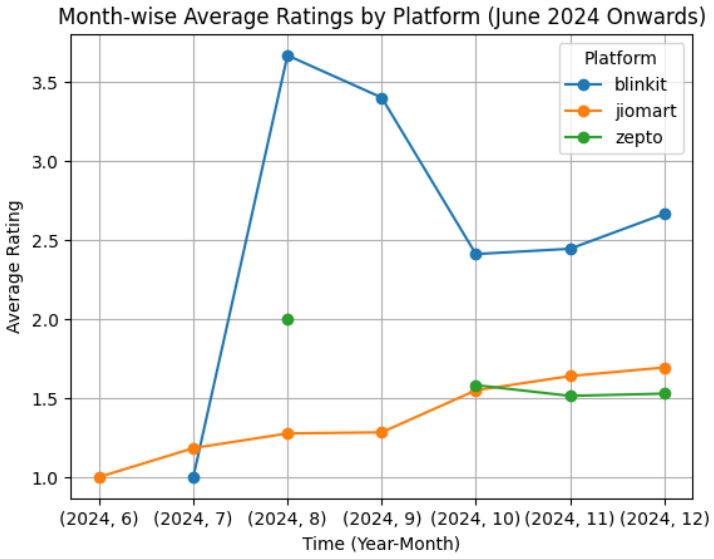
(code and results for text sentiment analysis):

```
# Ensure the date column is in datetime format
df['date'] = pd.to_datetime(df['date'])

# Group by platform and calculate the date range
date_range = df.groupby('platform')['date'].agg(['min', 'max']).reset_index()

# Rename columns for clarity (optional)
date_range.rename(columns={'min': 'start_date', 'max': 'end_date'}, inplace=True)

# Display the results
print(date_range)
```

```
platform start_date   end_date
0  blinkit 2024d07-27 2024-12-31
1  jiomart 2020-07-21 2024-12-31
2    zepto 2024-08-14 2024-12-31
```

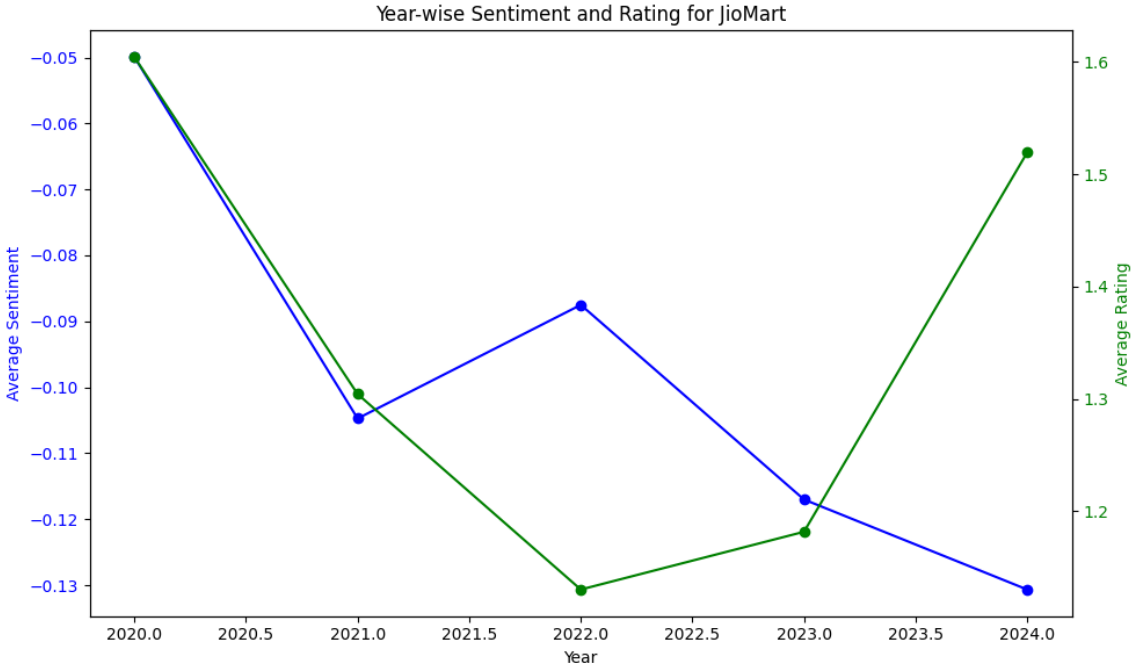
```
import pandas as pd
import matplotlib.pyplot as plt

# Step 1: Convert 'date' column to datetime
df['date'] = pd.to_datetime(df['date'])

# Step 2: Add necessary columns for grouping
df['year'] = df['date'].dt.year
df['month'] = df['date'].dt.month

# Step 3: Filter for data from June 2024 onwards
filtered_df = df[(df['year'] > 2024) | ((df['year'] == 2024) & (df['month'] >= 6))]

# Step 4: Group by platform, year, and month and calculate the average rating
grouped = filtered_df.groupby(['platform', 'year', 'month'])['rating'].mean().reset_index()

# Step 5: Pivot the data for easier plotting
pivot = grouped.pivot_table(index=['year', 'month'], columns='platform', values='rating')

# Step 6: Plot the data
```

```
plt.figure(figsize=(12, 6))
pivot.plot(marker='o')
plt.title('Month-wise Average Ratings by Platform (June 2024 Onwards)')
plt.xlabel('Time (Year-Month)')
plt.ylabel('Average Rating')
plt.legend(title='Platform')
plt.grid()
plt.show()
```




```
import pandas as pd
import matplotlib.pyplot as plt
from textblob import TextBlob

# Step 1: Filter Data for "JioMart"
df['platform'] = df['platform'].str.strip().str.lower()
# Now filter for 'jiomart'
jiomart_df = df[df['platform'] == 'jiomart']
print(jiomart_df)
#jiomart_df = df[df['platform'] == 'JioMart']

# Step 2: Perform Sentiment Analysis using TextBlob
def get_sentiment(text):
    # TextBlob sentiment polarity: 1 for positive, −1 for negative, 0 for neutral
    return TextBlob(str(text)).sentiment.polarity

jiomart_df['sentiment'] = jiomart_df['review'].apply(get_sentiment)

# Step 3: Extract the Year and calculate the average rating and sentiment by year
jiomart_df['year'] = jiomart_df['date'].dt.year
sentiment_yearwise = jiomart_df.groupby('year').agg(
    avg_sentiment=('sentiment', 'mean'),
    avg_rating=('rating', 'mean')
).reset_index()

# Step 4: Plot the Sentiment and Rating Graphs
fig, ax1 = plt.subplots(figsize=(10, 6))
```

```
# Plot the sentiment analysis
ax1.set_xlabel('Year')
ax1.set_ylabel('Average Sentiment', color='blue')
ax1.plot(sentiment_yearwise['year'], sentiment_yearwise['avg_sentiment'], color='blue', marker='o', label='Avg Sentiment')
ax1.tick_params(axis='y', labelcolor='blue')

# Create a second y-axis for the rating
ax2 = ax1.twinx()
ax2.set_ylabel('Average Rating', color='green')
ax2.plot(sentiment_yearwise['year'], sentiment_yearwise['avg_rating'], color='green', marker='o', label='Avg Rating')
ax2.tick_params(axis='y', labelcolor='green')

# Title and Display
plt.title('Year-wise Sentiment and Rating for JioMart')
fig.tight_layout()  # To prevent overlap
plt.show()
```

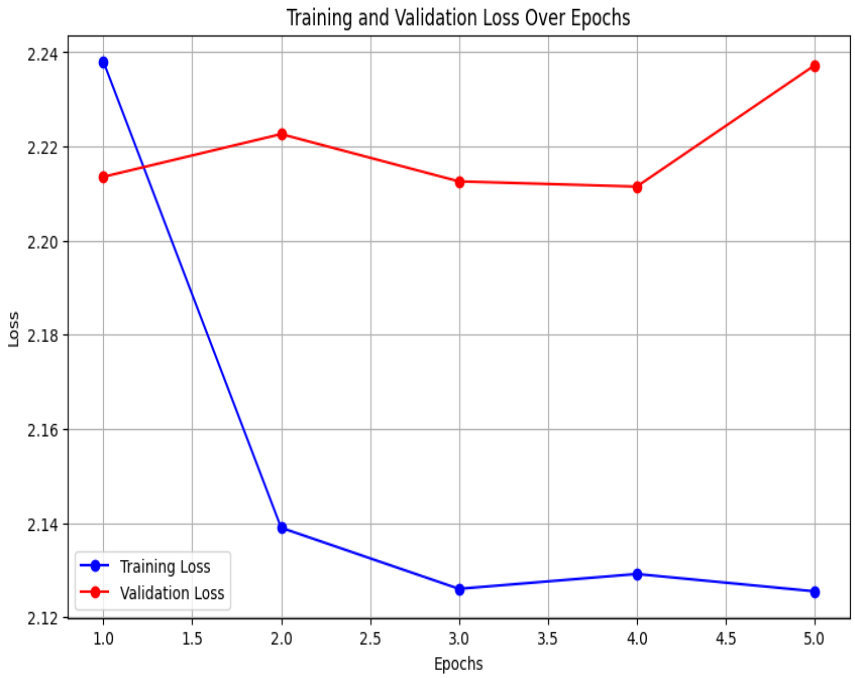


1. **Feature Extraction**:

LSTM for temporal dependencies in text (e.g., review sequences).

```
import numpy as np
import pandas as pd
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, LSTM, Dense, Dropout
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from sklearn.model_selection import train_test_split

# Load your dataset
#dfr = pd.read_csv("reviews.csv")

# Step 1: Preprocess text data
```

```
# Tokenize and pad sequences
tokenizer = Tokenizer(num_words=10000, oov_token="<OOV>")
tokenizer.fit_on_texts(df['review'])  # Assuming 'review' column contains text
sequences = tokenizer.texts_to_sequences(df['review'])
padded_sequences = pad_sequences(sequences, maxlen=200, padding='post', truncating='post')

# Prepare target variable
ratings = df['rating'].values  # Assuming 'rating' column contains ratings (0–5)

# Step 2: Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(padded_sequences, ratings, test_size=0.2, random_state=42)

# Step 3: Build the LSTM model
model = Sequential([
   Embedding(input_dim=10000, output_dim=128, input_length=200),  # Embedding layer
   LSTM(128, return_sequences=True),  # First LSTM layer
   Dropout(0.2),
   LSTM(64),  # Second LSTM layer
   Dense(32, activation='relu'),  # Fully connected layer
   Dense(1, activation='linear')  # Linear activation for regression output
])

# Compile the model
model.compile(optimizer='adam', loss='mse', metrics=['mae'])

# Step 4: Train the model
history = model.fit(
   X_train, y_train,
   epochs=5,
   batch_size=32,
   validation_data=(X_test, y_test)
)

# Step 5: Evaluate the model
test_loss, test_mae = model.evaluate(X_test, y_test)
print(f"Test Loss: {test_loss:.4f}, Test MAE: {test_mae:.4f}")

# Step 6: Predict on new data
sample_review = ["The product was fantastic! Highly recommend it."]
sample_sequence = tokenizer.texts_to_sequences(sample_review)
sample_padded = pad_sequences(sample_sequence, maxlen=200, padding='post', truncating='post')
predicted_rating = model.predict(sample_padded)
print(f"Predicted Rating: {predicted_rating[0][0]:.2f}")
```



CNN for spatial features in images.

Recurrent models or transformers for audio inputs.

Fusion Layer:

Combine feature vectors from each modality into a unified representation.

Classification:

Apply dense layers followed by a softmax activation function to classify sentiments into categories of rating from 1 to 5.

```
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input, Embedding, LSTM, Dense, Dropout, Flatten, Concatenate
from tensorflow.keras.applications import ResNet50
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences

# Define input dimensions
max_words = 5000
max_len = 100
embedding_dim = 100

# Text Input Pipeline
text_input = Input(shape=(max_len,))
embedding_layer = Embedding(max_words, embedding_dim, input_length=max_len)(text_input)
lstm_layer = LSTM(64, return_sequences=False)(embedding_layer)

# Image Input Pipeline
image_input = Input(shape=(224, 224, 3))
cnn_model = ResNet50(weights='imagenet', include_top=False, input_tensor=image_input)
cnn_output = Flatten()(cnn_model.output)

# Fusion
merged = Concatenate()([lstm_layer, cnn_output])
dense_1 = Dense(128, activation='relu')(merged)
dropout = Dropout(0.5)(dense_1)
```

```
output_layer = Dense(3, activation='softmax')(dropout)  # 3 classes for sentiment: Positive, Neutral, Negative

# Model
model = Model(inputs=[text_input, cnn_model.input], outputs=output_layer)

# Compile
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

# Summary
model.summary()
```

Appendix B

(Code for image and video sentiment analysis and combining with original code of text analysis as listed in Appendix A):

```
import cv2
import os
import numpy as np
import pandas as pd
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.applications.vgg16 import VGG16, preprocess_input
from tensorflow.keras.preprocessing.image import load_img, img_to_array
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input, Embedding, LSTM, Dense, Dropout, Concatenate, TimeDistributed
from tensorflow.keras.utils import to_categorical

# -------------------------------------
# Load dataset
# -------------------------------------
# Columns: 'text', 'image_path', 'video_path', 'label'
df = pd.read_csv("your_dataset.csv")

texts = df['text'].values
image_paths = df['image_path'].values
video_paths = df['video_path'].values
labels = df['label'].values

# -------------------------------------
# Text preprocessing
# -------------------------------------
max_vocab = 5000
max_len = 100

tokenizer = Tokenizer(num_words=max_vocab)
tokenizer.fit_on_texts(texts)
sequences = tokenizer.texts_to_sequences(texts)
text_data = pad_sequences(sequences, maxlen=max_len)
# -------------------------------------
# Image preprocessing
```

```
# ------------------------------------
def process_image(path):
    img = load_img(path, target_size=(224, 224))
    img = img_to_array(img)
    img = preprocess_input(img)
    return img

image_data = np.array([process_image(path) for path in image_paths])

# ------------------------------------
# Load VGG for image/video feature extraction
# ------------------------------------
vgg = VGG16(weights='imagenet', include_top=False, input_shape=(224, 224, 3))
for layer in vgg.layers:
    layer.trainable = False
vgg_model = Model(inputs=vgg.input, outputs=vgg.output)

# Extract image features
image_features = vgg_model.predict(image_data)
image_features = image_features.reshape(image_features.shape[0], -1)

# ------------------------------------
# Video feature extraction
# ------------------------------------
def process_video(video_path, max_frames=10):
    cap = cv2.VideoCapture(video_path)
    frames = []
    total_frames = int(cap.get(cv2.CAP_PROP_FRAME_COUNT))
    frame_interval = max(1, total_frames // max_frames)

    for i in range(max_frames):
        cap.set(cv2.CAP_PROP_POS_FRAMES, i * frame_interval)
        ret, frame = cap.read()
        if ret:
            frame = cv2.resize(frame, (224, 224))
            frame = preprocess_input(frame.astype(np.float32))
            frames.append(frame)
        else:
            break
    cap.release()

    frames = np.array(frames)
    return frames

video_sequences = []
max_video_frames = 10

for path in video_paths:
    frames = process_video(path, max_frames=max_video_frames)
    if frames.shape[0] < max_video_frames:
        # pad with zeros
        padding = np.zeros((max_video_frames - frames.shape[0], 224, 224, 3))
```

```
        frames = np.vstack((frames, padding))
    video_sequences.append(frames)

video_sequences = np.array(video_sequences)

# Extract frame-wise features
batch_size = video_sequences.shape[0]
video_features = []

for vid in video_sequences:
    features = vgg_model.predict(vid)
    features = features.reshape(features.shape[0], –1)
    video_features.append(features)

video_features = np.array(video_features)

# -------------------------------------
# Labels
# -------------------------------------
num_classes = len(np.unique(labels))
y = to_categorical(labels, num_classes=num_classes)

# -------------------------------------
# Model Building
# -------------------------------------

# Text branch
text_input = Input(shape=(max_len,))
embedding = Embedding(input_dim=max_vocab, output_dim=128)(text_input)
lstm_text = LSTM(64)(embedding)

# Image branch
image_input = Input(shape=(image_features.shape[1],))
dense_image = Dense(256, activation='relu')(image_input)

# Video branch
video_input = Input(shape=(max_video_frames, video_features.shape[2]))
lstm_video = LSTM(64)(video_input)

# Fusion
combined = Concatenate()([lstm_text, dense_image, lstm_video])
dropout = Dropout(0.5)(combined)
output = Dense(num_classes, activation='softmax')(dropout)

model = Model(inputs=[text_input, image_input, video_input], outputs=output)
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

# -------------------------------------
# Training
# -------------------------------------
model.fit(
    [text_data, image_features, video_features],
```

```
    y,
    epochs=10,
    batch_size=16,
    validation_split=0.2
)
```

Appendix C

Code for Evaluation of performance:

```
import pandas as pd
from collections import Counter
from nltk.corpus import stopwords
import re
import nltk

# Download stopwords if not already downloaded
nltk.download('stopwords')

# Step 1: Filter reviews by rating
rating_5_reviews = df[df['rating'] == 5]['review']
rating_1_to_4_reviews = df[df['rating'] < 5]['review']

# Step 2: Tokenize and clean the reviews
def tokenize_and_clean(reviews):
    stop_words = set(stopwords.words('english'))
    all_words = []
    for review in reviews:
        # Lowercase and remove non-alphanumeric characters
        tokens = re.findall(r'\b\w+\b', str(review).lower())
        # Remove stopwords
        tokens = [word for word in tokens if word not in stop_words]
        all_words.extend(tokens)
    return set(all_words)  # Return a set of unique words

words_rating_5 = tokenize_and_clean(rating_5_reviews)
words_rating_1_to_4 = tokenize_and_clean(rating_1_to_4_reviews)

# Step 3: Find unique words in rating 5 reviews
unique_words_rating_5 = words_rating_5 - words_rating_1_to_4

# Step 4: Count frequencies of unique words in rating 5 reviews
def count_word_frequencies(reviews, unique_words):
    word_counter = Counter()
    for review in reviews:
        tokens = re.findall(r'\b\w+\b', str(review).lower())
        for word in tokens:
            if word in unique_words:
                word_counter[word] += 1
    return word_counter.most_common(10)

top_words = count_word_frequencies(rating_5_reviews, unique_words_rating_5)
```

```
# Display the 10 most common words
print("10 Most Common Words with Rating 5:")
for word, freq in top_words:
   print(f"{word}: {freq}")
10 Most Common Words with Rating 5:
wonderful: 16
lightning: 12
outstanding: 10
lifesaver: 8
incredible: 8
superfast: 8
changer: 7
exceptional: 6
fabulous: 6
blowing: 5

The following code helps to determine the confusion metics.
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
import numpy as np

# Round the predicted ratings to the nearest integer
y_pred = model.predict(X_test)
y_pred_rounded = np.rint(y_pred).astype(int)  # Round to nearest integer

# Ensure predicted ratings stay within valid bounds (0 to 5)
y_pred_rounded = np.clip(y_pred_rounded, 0, 5)

# Round the true labels to integers (optional if they are not floats)
y_test_rounded = np.rint(y_test).astype(int)

# Compute the confusion matrix
cm = confusion_matrix(y_test_rounded, y_pred_rounded, labels=[0, 1, 2, 3, 4, 5])

# Plot the confusion matrix
disp = ConfusionMatrixDisplay(confusion_matrix=cm, display_labels=[0, 1, 2, 3, 4, 5])
disp.plot(cmap='Blues')
plt.title('Confusion Matrix')
plt.show()
```

References

1. Yu, Y.; Si, X.; Hu, C.; et al. A review of recurrent neural networks: LSTM cells and network architectures. *Neural Comput* **2019**, *31*, 1235–1270. [CrossRef]
2. Sherstinsky, A. Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network. *Physica D* **2020**, *404*, 132306. [CrossRef]
3. Bansal, N.; Singla, B. Predictive modeling for event-free survival in allogeneic hematopoietic stem cell transplantation patients using LSTM networks. *Foundry* **2025**, *28*, 55–61.
4. Chen, K.; Wang, R.; Utiyama, M.; et al. Content Word Aware Neural Machine Translation. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, Stroudsburg, PA, United States, 5–10 July 2020. pp. 358–364.
5. Tang, D.; Qin, B.; Liu, T. Document modeling with gated recurrent neural network for sentiment classification. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, Lisbon, Portugal, 17–21 September 2015. pp. 1422–1432.

6. Wankhade, M.; Rao, A.C.S.; Kulkarni, C.; et al. A survey on sentiment analysis methods, applications, and challenges. *Artif. Intell. Rev.* **2022**, *55*, 5731–5780. [CrossRef]
7. Hussein, D.M.E.D.M. A survey on sentiment analysis challenges. *J. King Saud Univ.-Eng. Sci.* **2018**, *30*, 330–338. [CrossRef]
8. Zhu, L.; Zhu, Z.; Zhang, C.; et al. Multimodal sentiment analysis based on fusion methods: A survey. *Information Fusion* **2023**, *95,* 306–325. [CrossRef]
9. Ma, Y.; Peng, H.; Khan, T.; et al. Sentic LSTM: a hybrid network for targeted aspect-based sentiment analysis. *Cogn. Comput.* **2018**, *10*, 639–650. [CrossRef]
10. Mikolov, T.; Chen, K.; Corrado, G.; et al. Efficient estimation of word representations in vector space. *arXiv* **2013**, *revised*. arXiv:1301.3781 [cs.CL]. [CrossRef]
11. Mikolov, T.; Sutskever, I.; Chen, K.; et al. Distributed representations of words and phrases and their compositionality. *Adv. Neural Inf. Process. Syst.* **2013**, *26*, 3111–3119.
12. Pennington, J.; Socher, R.; Manning, C. GloVe: global vectors for word representation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Doha, Qatar, 25–29 October 2014. pp. 1532–1543.
13. Anoop, V.S.; Asharaf, S. Aspect-oriented sentiment analysis: A topic modeling-powered approach. *J. Intell. Syst* **2019**, *29*, 1166-1178. [CrossRef]
14. Wu, C.; Ma, B.; Zhang, Z.; et al. Evaluating Zero-Shot Multilingual Aspect-Based Sentiment Analysis with Large Language Models. *arXiv* **2024**, *preprint arXiv:2412.12564*.
15. Shen, L.; Feng, Y.; Zhan, H. Modeling Semantic Relationship in Multi-turn Conversations with Hierarchical Latent Variables. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, Florence, Italy, 28 July–2 August 2019. pp. 5497–5502.
16. Singh, A.; Thakur, N.; Sharma, A. A review of supervised machine learning algorithms. In Proceedings of 2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom), New Delhi, India, 16–18 March 2016. pp. 1310–1315.
17. Osisanwo, F.Y.; Akinsola, J.E.T.; Awodele, O.; et al. Supervised machine learning algorithms: classification and comparison. *Int. J. Comput. Trends Technol.* **2017**, *48*, 128–138. [CrossRef]
18. Poria, S.; Cambria, E.; Hazarika, D.; et al. A Deeper Look into Sarcastic Tweets Using Deep Convolutional Neural Networks. In Proceedings of the COLING 2016, the 26th International Conference on Computational Linguistics, Osaka, Japan, 11–16 December 2016. pp. 1601–1612.
19. Zadeh, A.; Zellers, R.; Pincus, E.; et al. Multimodal sentiment intensity analysis in videos: Facial gestures and verbal messages. *IEEE Intell. Syst.* **2016**, *31*, 82–88. [CrossRef]
20. Singla, B. Fine-tuning FaceNet for celebrity face Recognition: Enhancing Real vs. Fake Image Detection on the '105_Classes_Pins' Dataset. *Degres* **2024**, *9*, 117–129.
21. Singla, B.; Verma, A.K. Application of Machine Learning for the Tracing of Jellyfish Attack in MANETs. In *Science and Technology: Developments and Applications*; Jakóbczak, D.J. Ed.; BP International: London, United Kingdom, 2025; pp. 50–65.
22. Zhou, C.; Sun, C.; Liu, Z.; et al. A C-LSTM neural network for text classification. *arXiv* **2015**, arXiv preprint arXiv:1511.08630. [CrossRef]
23. Huang, H., Jin, Y., Rao, R. Sentiment-aware transformer using joint training. In Proceedings of 2020 IEEE 32nd International Conference on Tools with Artificial Intelligence (ICTAI), Baltimore, MD, USA, 9–11 November 2020.
24. Liu, Y.; Ott, M.; Goyal, N.; et al. RoBERTa: A robustly optimized BERT pretraining approach. *arXiv* **2019**, arXiv preprint arXiv:1907.11692. [CrossRef]
25. Han, Y., Moghaddam, M. Design knowledge as attention emphasizer in large language model-based sentiment analysis. *J. Comput. Inf. Sci. Eng.* **2025**, *25(2)*, 021007. [CrossRef]
26. Wang, Y.; Huang, M.; Zhu, X.; et al. Attention-based LSTM for aspect-level sentiment classification. In Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, 1–5 November 2016, Austin, Texas, USA. pp. 606–615. [CrossRef]