Uber Reviews Sentiment Analysis Using Forecasting

202418009[1], **202418011**[2], **202418048**[3]

202418009@daiict.ac.in
202418011@daiict.ac.in
202418048@daiict.ac.in

Problem Formulation

:- Understanding sentiment trends helps businesses across industries—from gauging campaign impact to improving customer experience. Analyzing early feedback offers insights into audience behavior, while forecasting future opinions enables companies to proactively meet customer needs and boost satisfaction.

The methods we can use include sentiment analysis with time series forecasting. By doing so, we can identify existing patterns in the data and understand how user sentiments evolve over time. Textual sentiment can reveal nuanced user feelings (e.g., "Very convenient" vs. "Good") that raw scores might miss, helping organizations understand specific pain points or strengths.

The goal of this research is to **Forecast Sentiment Based on Textual Review Content**. The objective of this model is to **predict future sentiment trends** by analyzing the evolution of textual sentiment in the content column over time.

1.1 Sentiment Classification using BERT

Let $D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$ represent a dataset of n user reviews collected from the Uber platform, where each x_i corresponds to the textual content of the i-th review, and $y_i \in \{0, 1, 2\}$ is the sentiment label indicating whether the review is **negative** (0), **neutral** (1), or **positive** (2). The goal of this stage is to accurately classify each review into one of the three sentiment categories.

To achieve this, we utilize the pre-trained **Bidirectional Encoder Representations from Transformers (BERT)** model, specifically the `bert-base-uncased` variant. BERT is a deep transformer-based architecture that captures both left and right context in all layers through its bidirectional self-attention mechanism. Each input review x_i is first tokenized and then passed through the BERT model to obtain a contextualized embedding of the entire input:

$$\mathbf{h}_i = f_{\text{BERT}}(x_i) \in \mathbb{R}^d$$

Here, $\mathbf{h}_i$ represents the output vector (typically the `[CLS]` token embedding) from the final layer of BERT, and d is the hidden size of the model (usually $d = 768$ for the base model). This embedding captures the semantic meaning of the entire review based on its context.

To classify the sentiment, this representation is passed through a task-specific classification head—a fully connected linear layer—followed by a softmax activation to produce the probability distribution over the three sentiment classes:

$$\hat{y}_i = \text{softmax}(W\mathbf{h}_i + \mathbf{b})$$

where $W \in \mathbb{R}^{3\times d}$ and $\mathbf{b} \in \mathbb{R}^3$ are learnable parameters optimized during fine-tuning.

The model is trained using the categorical cross-entropy loss, defined as:

$$\mathcal{L}_{\text{CE}} = -\sum_{i=1}^{n} y_i \log(\hat{y}_i)$$

This loss function penalizes the model more when its predicted probability for the correct class is low, guiding it to improve accuracy over training epochs. Fine-tuning the pre-trained BERT model on the Uber review dataset allows the model to adapt its learned language representations to the domain-specific sentiment classification task.

1.2 Sentiment Time Series Construction

After obtaining the sentiment predictions for each individual review, the next step involves aggregating these sentiments over fixed temporal intervals to construct a time series representation. This time series forms the basis for analyzing trends in user sentiment over time and serves as the foundation for sentiment forecasting.

We define the sentiment time series as:

$$S = \{s_1, s_2, \ldots, s_T\}$$

where s_t denotes the aggregated sentiment value for the time interval t, such as a day or a week, and T is the total number of time periods. Each s_t can be computed using one of several strategies:

- **Mean Score Aggregation:** Assign numerical values to sentiment labels (e.g., 0 = negative, 1 = neutral, 2 = positive) and compute the average sentiment score per time unit.

- **Proportional Aggregation:** Calculate the proportion of each sentiment class within each interval and use it to model sentiment dynamics.
- **Weighted Aggregation:** Optionally apply weights to reviews based on factors such as review length or user credibility.

The resulting time series S reflects how public sentiment evolves over time, providing a structured signal that can be used for predictive modeling. This approach also aids in visualizing peaks and dips in user satisfaction, which may correlate with events such as service changes, promotions, or external influences (e.g., weather conditions, holidays).

1.3 Sentiment Forecasting using GRU

Once the sentiment time series has been constructed, the final stage involves forecasting future sentiment values. Forecasting allows platforms like Uber to proactively address potential user dissatisfaction or capitalize on emerging positive trends.

We denote the forecasting input at time t as a sequence of past k sentiment values:

$$X_t = [s_{t-k}, s_{t-k+1}, \ldots, s_{t-1}]$$

and the target output as the sentiment value s_t. The objective is to learn a function that maps the historical window X_t to the future sentiment prediction $\hat{s}_t$.

To achieve this, we employ a **Gated Recurrent Unit (GRU)** based neural network. GRUs are a type of Recurrent Neural Network (RNN) designed to capture temporal dependencies in sequential data while mitigating issues such as vanishing gradients. GRUs utilize two primary gates—the reset gate and the update gate—which control the flow of information and determine how much of the past should be retained or forgotten.

The GRU model is trained to minimize the Mean Squared Error (MSE) between the predicted and actual sentiment values:

$$L_{MSE} = \frac{1}{T-k} \sum_{t=k+1}^{T} (s_t - \hat{s}_t)^2$$

This loss function ensures that the model learns to produce predictions that closely match the observed sentiment values. In our implementation, the GRU model is trained to predict sentiment trends up to 7 days into the future, enabling short-term sentiment forecasting.

This forecasting capability is crucial for operational planning, marketing strategies, and enhancing customer experience by anticipating and responding to sentiment dynamics.

By integrating BERT-based sentiment classification with GRU-based time series forecasting, our framework not only interprets current user sentiment but also provides valuable foresight into its future trajectory. This enables a data-driven, proactive approach to sentiment management.

Implementation Details

This section elaborates on the complete implementation pipeline, encompassing dataset characteristics, preprocessing techniques, model training configurations, time series construction, forecasting architecture, and tools used throughout the study.

2.1 Dataset and Preprocessing

The primary dataset employed in this research comprises 12,000 Uber user reviews, each accompanied by relevant metadata such as review content, date of submission, app version, and user rating scores. The dataset was sourced in raw textual format and required rigorous preprocessing to make it suitable for natural language processing (NLP) and deep learning models.

To clean and normalize the data, a custom preprocessing pipeline was developed. The following steps were applied sequentially to the raw review text:

1. **Lowercasing:** All characters were converted to lowercase to ensure uniformity and reduce redundancy caused by case sensitivity.
2. **Punctuation Removal:** All non-alphanumeric characters, including punctuation marks, were removed using regular expressions to reduce noise.
3. **Emoji Handling:** Emojis and emoticons, which often carry sentiment information, were converted into textual descriptions using the `emoji.demojize()` function to preserve their semantic meaning.
4. **Tokenization:** Text was tokenized into individual words using `nltk.word_tokenize`, allowing word-level manipulation.
5. **Lemmatization and Stemming:** Each token was lemmatized using `WordNetLemmatizer` and then stemmed using `PorterStemmer` to reduce vocabulary size and enhance generalization.

Stop words and extremely short reviews (e.g., one-word responses) were removed to maintain data quality. The final result was a clean, tokenized, and normalized corpus, suitable for input into the sentiment classification model.

2.2 Sentiment Classification Using BERT

For sentiment classification, the study utilized the `bert-base-uncased` model from the HuggingFace `transformers` library. BERT (Bidirectional Encoder Representations from Transformers) is a transformer-based language model pre-trained on a large corpus of English text. Fine-tuning was performed on the Uber review dataset to adapt the model for domain-specific sentiment classification.

Model Training Configuration:

- **Model:** BERT-base-uncased (12 transformer layers, 768 hidden size, 12 attention heads)
- **Batch Size:** 16
- **Max Sequence Length:** 128 tokens
- **Learning Rate:** 2e-5
- **Epochs:** 5

- **Optimizer:** AdamW
- **Loss Function:** Categorical Cross-Entropy

The dataset was split using stratified sampling into 80% training and 20% testing sets to preserve class balance. Sentiment labels were mapped numerically as follows: 0 = Negative, 1 = Neutral, and 2 = Positive. Each review was tokenized using the BERT tokenizer, with sequences padded or truncated to the maximum length.

Model performance was evaluated using standard metrics: accuracy, precision, recall, and F1-score for each sentiment category. After training, the model produced sentiment class probabilities using a softmax activation over the three classes.

2.3 Time Series Aggregation

To analyze the temporal dynamics of user sentiment, the model-generated sentiment predictions were aggregated over time to construct a multivariate sentiment time series. For each date in the dataset, sentiment predictions were grouped, and the following statistics were computed:

- Mean Score of Positive Sentiments
- Mean Score of Neutral Sentiments
- Mean Score of Negative Sentiments

These values were calculated using the softmax probabilities output by the BERT model. This aggregation process transformed a sequence of individual sentiment predictions into a structured time series dataset:

$$S_t = \{\text{score}_{\text{positive}}, \text{score}_{\text{neutral}}, \text{score}_{\text{negative}}\}_t$$

Each time step t in the series represented a calendar day, and each component in S_t reflected the average sentiment distribution for that day. This time series provided a foundation for sentiment forecasting and trend analysis.

2.4 Sentiment Forecasting Using GRU

To anticipate future sentiment trends, a Gated Recurrent Unit (GRU)-based neural network was designed and trained. GRUs are a variant of recurrent neural networks (RNNs) that efficiently capture temporal dependencies while mitigating issues like vanishing gradients. The GRU model was constructed to predict user sentiment scores for the next 7 days based on the previous 15 days of observed data.

GRU Model Architecture:

- **Input Size:** 3 (aggregated positive, neutral, negative sentiment scores)
- **Hidden Layer Size:** 32
- **Sequence Input Length:** 15 (time steps representing the past 15 days)
- **Forecast Horizon:** 7 days
- **GRU Layers:** 2
- **Dropout Rate:** 0.3
- **Output Layer:** Flattened fully connected layer producing 21 values (7 days × 3 sentiment types)

Training Configuration:

- **Epochs:** 50 (with early stopping based on validation loss; patience = 5)
- **Batch Size:** 16
- **Optimizer:** Adam
- **Initial Learning Rate:** 0.001
- **Loss Function:** Mean Squared Error (MSE)
- **Learning Rate Scheduler:** ReduceLROnPlateau (factor = 0.5, patience = 2)

Before training, the time series data was normalized using MinMax scaling to bring all values within the range [0, 1], ensuring stable convergence. The training and validation split was maintained at 80/20. After training, the predicted outputs were inverse-scaled to restore the original sentiment score range. For each day in the forecast horizon, the dominant sentiment (i.e., the class with the highest score among the three) was interpreted as the predicted sentiment class.

This forecasting framework enabled short-term modeling of sentiment dynamics and allowed for the detection of emerging positive or negative trends in user perception.

2.5 Tools and Environment

The entire implementation was carried out using the Python programming language in a cloud-based environment provided by Google Colab, which included free access to NVIDIA GPUs for accelerated model training.

Key Libraries and Frameworks:

- **PyTorch** – Deep learning framework used for model development and training.
- **HuggingFace Transformers** – Library used for leveraging pre-trained BERT models and tokenizers.
- **Scikit-learn** – Used for evaluation metrics, preprocessing, and model utilities.
- **NLTK (Natural Language Toolkit)** – Employed for text preprocessing tasks including tokenization, lemmatization, and stemming.
- **Pandas and NumPy** – Used for data manipulation and numerical computations.
- **Matplotlib and Seaborn** – Used for data visualization and plotting sentiment trends over time.

The models and datasets were versioned and saved within the Colab environment, and trained models were exported for downstream inference and evaluation.

Model Architecture

The architecture employed in this study is composed of two major components: (1) a sentiment classification model based on a pre-trained transformer model (BERT), and (2) a time series forecasting model using a Gated Recurrent Unit (GRU) architecture. This two-stage pipeline enables both the extraction of fine-grained sentiment insights from textual reviews and the modeling of their temporal dynamics for future prediction.

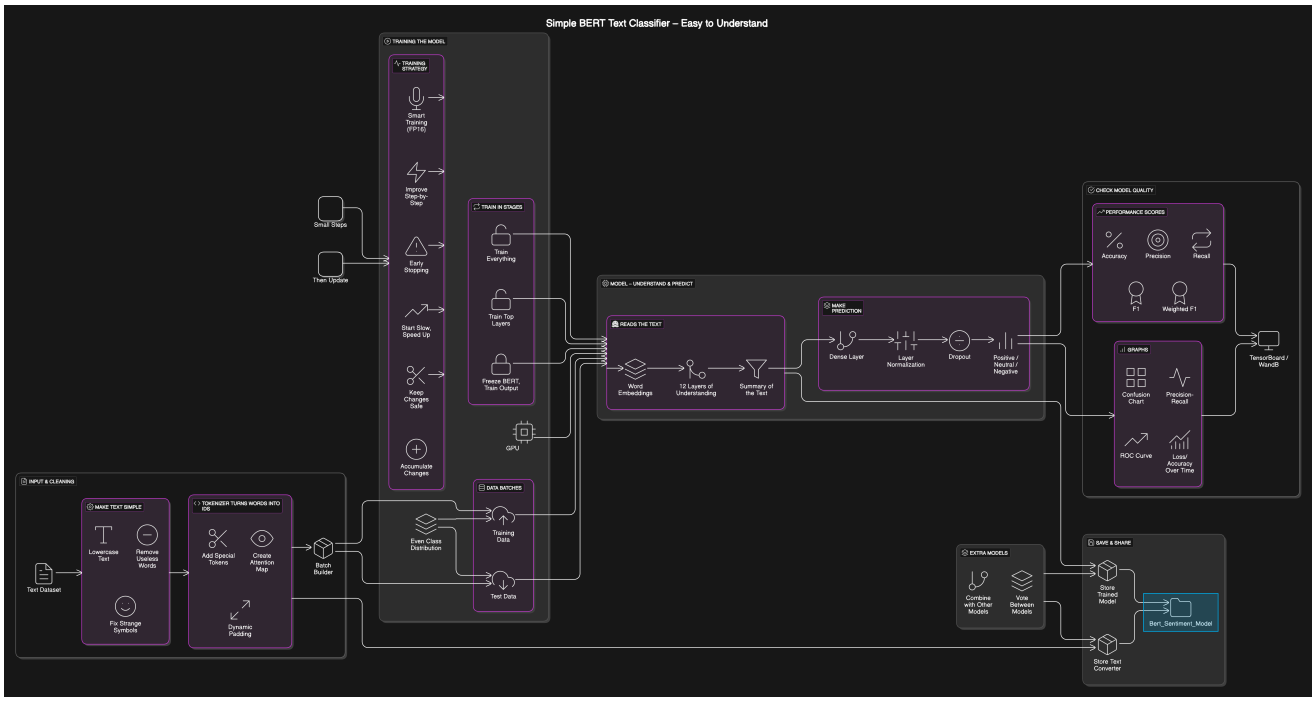

Figure 1: BERT Architecture. BERT leverages the Transformer's encoder mechanism to capture contextual relationships in both directions (left-to-right and right-to-left) for pretraining on massive text corpora, excelling in understanding language semantics.

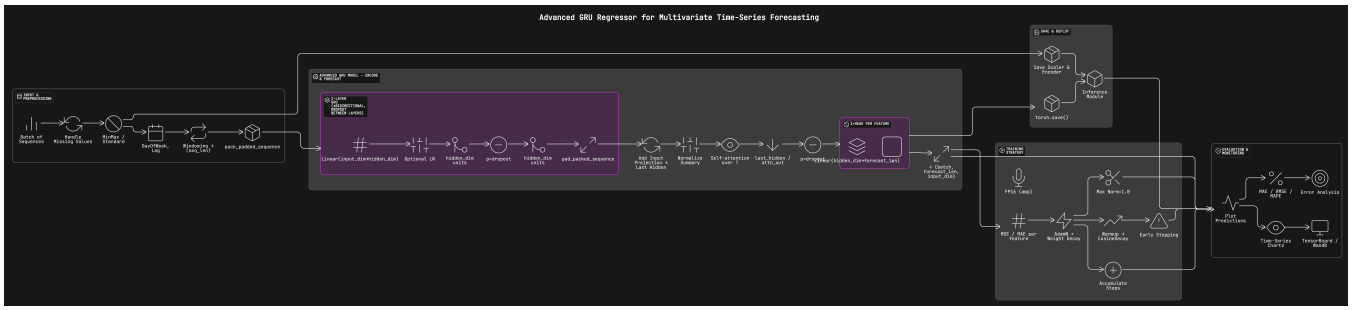

Figure 2: GRU architecture. The plot indicates that the GRU architecture simplifies the RNN by combining the forget and input gates into a single update gate, making it efficient for sequence modeling tasks with fewer parameters than LSTMs.

3.1 Sentiment Classification using BERT

The first stage of the architecture is designed to perform sentiment classification on Uber user reviews. For this task, we utilized the `bert-base-uncased` model from the HuggingFace Transformers library, a widely adopted pre-trained variant of Bidirectional Encoder Representations from Transformers (BERT). BERT is a deep bidirectional transformer model developed by Google, pre-trained on a large corpus (BooksCorpus and English Wikipedia) using masked language modeling and next sentence prediction objectives.

The `bert-base-uncased` variant used in our experiments consists of:

- 12 transformer encoder layers (blocks)
- 12 self-attention heads per layer
- A hidden size of 768
- 110 million trainable parameters

Each input review text was first tokenized using the BERT tokenizer, which converts raw text into WordPiece tokens. A special `[CLS]` token was added at the beginning of each input sequence to represent the aggregate embedding of the sequence, and a `[SEP]` token was appended to mark the end. The tokenized sequence was padded or truncated to a maximum length of 128 tokens to ensure uniform input size.

The tokenized input (including `input_ids` and `attention_mask`) was then passed through the transformer encoder layers of BERT. The output corresponding to the `[CLS]` token (i.e., the first token's output vector of size 768) was used as a contextualized embedding representing the entire input sentence. This embedding was fed into a classification head, which consisted of a fully connected linear layer projecting the 768-dimensional vector

to a 3-dimensional output, corresponding to the sentiment categories: negative (0), neutral (1), and positive (2).

The final output logits were passed through a softmax activation function to generate normalized class probabilities. The model was fine-tuned using cross-entropy loss and the AdamW optimizer, with hyperparameters tuned for optimal performance:

- Learning Rate: 2×10^{-5}
- Batch Size: 16
- Epochs: 5

During training, BERT's transformer layers were unfrozen, allowing the model to adapt its pre-trained knowledge to the domain-specific language and sentiment expressions in the Uber reviews dataset.

3.2 Sentiment Time Series Forecasting using GRU

The second stage of the system focused on capturing and forecasting the temporal evolution of user sentiment over time. Once sentiment predictions were generated for each review, they were aggregated on a daily basis to produce a multivariate sentiment time series, with each day represented by a vector of average sentiment scores for positive, neutral, and negative sentiments.

To forecast sentiment trends for the next 7 days, a Gated Recurrent Unit (GRU)-based neural network was employed. GRUs are a type of recurrent neural network (RNN) that offer a simplified alternative to Long Short-Term Memory (LSTM) networks, while maintaining the ability to capture long-term dependencies in sequential data. GRUs are particularly effective for time series prediction due to their efficient gating mechanisms.

The input to the GRU model was a sequence of 15 consecutive days of aggregated sentiment scores, represented as a matrix of shape $(15, 3)$—with each row corresponding to a day and each column representing the average score for one of the three sentiment classes.

GRU Model Architecture:

- Two stacked GRU layers with a hidden size of 32
- Dropout layer with a dropout rate of 0.3
- Fully connected output layer projecting to a 21-dimensional vector (7 days $\times$ 3 sentiments)

Mathematically, the GRU learns a function $f_{GRU}(X_t)$ to predict future sentiment scores $\hat{s}_{t+1}, \hat{s}_{t+2}, \ldots, \hat{s}_{t+7}$, where $X_t \in \mathbb{R}^{15\times 3}$ is the input sequence at time step t. The model was trained to minimize the Mean Squared Error (MSE) between the predicted and actual sentiment scores. The data was normalized using MinMax scaling and inverse-transformed after prediction for interpretability.

Training Configuration:

- Epochs: 50 (early stopping with patience = 5)
- Batch Size: 16
- Optimizer: Adam
- Learning Rate: 0.001
- Learning Rate Scheduler: ReduceLROnPlateau (factor = 0.5, patience = 2)

This architecture enables the GRU network to capture the temporal correlations and trends in sentiment data. The final output provides short-term sentiment forecasts, supporting proactive decision-making in customer experience and service improvement.

Baseline Models

To assess the performance of the proposed BERT-GRU framework, baseline models were implemented for both sentiment classification and forecasting tasks.

4.1 Sentiment Classification Baselines

For sentiment classification, two popular lexicon-based methods were used: **VADER** and **TextBlob**. These models rely on predefined sentiment dictionaries and do not require training on labeled data.

- **VADER** demonstrated strong performance on the Uber review dataset, achieving precision scores of 0.98 (negative), 0.97 (neutral), and 1.00 (positive), with an overall F1-score of 0.99. It was particularly effective on structured text.
- **TextBlob**, though fast and lightweight, showed reduced effectiveness, especially with informal language. It achieved an overall accuracy of 78%, and F1-scores of 0.63 (negative), 0.63 (neutral), and 0.88 (positive).

While both methods are useful for quick sentiment estimates, they lack contextual understanding. The fine-tuned BERT model, leveraging deep contextual embeddings, significantly outperformed both by handling nuanced and domain-specific language more effectively.

4.2 Sentiment Forecasting Baseline

For forecasting sentiment trends, a **bi-directional Long Short-Term Memory (biLSTM)** network was implemented as a baseline. Trained on the same aggregated daily sentiment time series, it used 15 days of past data to forecast the next 7 days.

biLSTM Performance Metrics:

- Mean Squared Error (MSE): 0.0001
- Mean Absolute Error (MAE): 0.0074
- Root Mean Squared Error (RMSE): 0.0091

Despite its strong performance, the GRU-based model used in our framework outperformed it by training faster and yielding lower validation errors. GRU's simpler architecture made it more computationally efficient while maintaining or improving accuracy.

Conclusion: While traditional baselines provided a solid foundation, the BERT-GRU architecture delivered superior results in both classification and forecasting, demonstrating the value of combining transformer-based understanding with temporal sequence modeling.

Experiment Setup

This section outlines the comprehensive experimental workflow employed to assess the effectiveness of different models

for sentiment classification and time series forecasting based on user reviews from the Uber platform. The experiments were structured to compare the performance of lexicon-based models and deep learning-based models in terms of classification accuracy and forecasting precision. The entire pipeline—from data acquisition and preprocessing to model training, evaluation, and saving—was implemented using Python-based tools in a GPU-enabled environment.

5.1 Dataset and Preprocessing

The dataset consisted of 12,000 Uber user reviews collected from app stores, containing essential metadata such as review text, submission date, application version, and sentiment labels categorized into three classes: negative, neutral, and positive.

Prior to modeling, the text data underwent extensive preprocessing to normalize and standardize the content. The preprocessing pipeline included:

- **Lowercasing**: All characters converted to lowercase to remove case-based variance.
- **Punctuation Removal**: Eliminated irrelevant symbols.
- **Emoji Conversion**: Emojis transformed into textual descriptions using the `emoji.demojize()` function.
- **Tokenization**: Split sentences into tokens using `nltk.word_tokenize`.
- **Lemmatization and Stemming**: Applied `WordNetLemmatizer` and `PorterStemmer` to reduce words to their base and root forms.

This ensured a clean, semantically consistent input for both rule-based and deep learning models.

5.2 Baseline Models

To establish reference points, we implemented the following baselines:

- **VADER**: A rule-based sentiment analyzer tailored for social media text. Utilizes sentiment lexicons and syntactic heuristics. Demonstrated strong performance on reviews with clear sentiment markers.
- **TextBlob**: Lightweight sentiment analyzer that outputs polarity and subjectivity scores. While easy to implement, it showed limited effectiveness on informal user-generated content.
- **LSTM (Long Short-Term Memory)**: Used for time series forecasting. Trained on 15 days of sentiment data to predict the next 7. Despite reasonable accuracy, it was slower and more complex than GRU.

5.3 Advanced Deep Learning Models

To overcome limitations of the baselines, we deployed advanced models:

BERT for Sentiment Classification We fine-tuned a pretrained `bert-base-uncased` model from HuggingFace with the following configuration:

- 12 transformer encoder layers, 12 attention heads, hidden size of 768
- Batch size: 16, Sequence length: 128, Learning rate: $2e{-}5$
- Epochs: 5, Optimizer: AdamW, Loss: Cross-Entropy

Each review was tokenized and padded to uniform length. The [CLS] token's final-layer embedding was passed through a linear layer and softmax to produce sentiment class probabilities. Evaluation metrics included accuracy, precision, recall, and F1-score. BERT consistently outperformed lexicon-based methods due to its contextual understanding.

GRU for Time Series Forecasting To predict sentiment for the next 7 days, we used a GRU-based model trained on 15-day sentiment sequences. Inputs were MinMax-normalized. Configuration:

- Input size: 3 (positive, neutral, negative), Hidden size: 32, Sequence length: 15
- GRU layers: 2, Dropout: 0.3, Output: 21 values (7 days $\times$ 3 sentiments)
- Epochs: 50, Batch size: 16, Optimizer: Adam, LR: 0.001
- Loss: Mean Squared Error (MSE), LR Scheduler: ReduceLROnPlateau

Early stopping (patience = 5) was applied. The GRU model was more efficient than LSTM and delivered comparable or better forecasting accuracy.

5.4 Training and Evaluation Environment

Models were trained and evaluated on an 80/20 stratified split using a Google Colab GPU instance. Libraries used:

- **PyTorch**: Model training and inference
- **HuggingFace Transformers**: BERT model integration
- **Scikit-learn**: Preprocessing and metrics
- **NLTK**: Text normalization and tokenization
- **Matplotlib & Seaborn**: Trend visualization

Model checkpoints were saved throughout training. The best-performing models were preserved for further analysis and deployment.

Citation and References

1. Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8), 1735-1780. https://doi.org/10.1162/neco.1997.9.8.1735
2. Loria, S. (2018). *TextBlob: Simplified Text Processing*. Retrieved from https://textblob.readthedocs.io/en/dev/
3. Cho, K., Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. *Proceedings of EMNLP 2014*, 1724-1734. https://aclanthology.org/D14-1179
4. Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. *Proceedings of NAACL-HLT 2019*, 4171-4186. https://doi.org/10.18653/v1/N19-1423

Future Scope

The future scope of the research can focus on several directions to enhance the model's performance, extend its applications, and explore new areas of improvement:

- **Exploring Other Transformer Variants:** While BERT is highly effective, other variants like RoBERTa, ALBERT, DistilBERT, and XLNet can be explored for the same task. These models are optimized to be faster or more efficient and might outperform BERT in specific cases.
- **Integrating External Data:** Integrating additional external data sources such as user behavior data, transaction data, or social media data can enhance sentiment analysis models, leading to more accurate predictions.
- **Model Interpretability:** Increasing the interpretability of models, particularly deep learning models like BERT and GRU, can be an area of future research. Techniques like attention visualization or LIME (Local Interpretable Model-Agnostic Explanations) can help better understand how these models are making decisions.
- **Multi-task Learning:** Another area to explore is multi-task learning, where the model can learn multiple tasks simultaneously, such as sentiment analysis, classification, and time-series forecasting. This can help in leveraging shared knowledge across tasks.
- **GRU for Other Forecasting Tasks:** Besides the 7-day forecasting, GRU can be applied to other types of time-series data for more complex forecasting tasks, such as stock prices, temperature predictions, etc. Expanding GRU to handle more varied forecasting tasks could significantly extend its application.

Limitations

- **Data Quality and Bias:** The performance of the models heavily depends on the quality and the diversity of the dataset used for training. If the data is biased or not representative of the entire population, the model's performance could degrade. For instance, if the sentiment labels are imbalanced, it may affect the generalizability of the model.
- **Model Interpretability:** While BERT and GRU are powerful models, they are often considered "black boxes," meaning it's challenging to interpret how exactly they make predictions. This can be a limitation when transparency and explainability are required in critical applications.
- **Computational Resources:** Using models like BERT requires substantial computational power, especially for training the models from scratch. In addition, BERT models can be memory-intensive and slow for inference, particularly when processing large datasets.
- **Overfitting in LSTM/GRU for Forecasting:** Time-series models like LSTM/GRU can suffer from overfitting, especially if the model is too complex relative to the amount of available data. This overfitting can lead to poor generalization to unseen data.
- **Generalization to Other Domains:** Although the sentiment analysis model is evaluated on a specific dataset (e.g., Uber reviews), there may be challenges in generalizing the model to different domains. Domain adaptation and transfer learning could address this limitation, but it remains a challenge.
- **Forecasting in Noisy Data:** GRU-based models for forecasting time-series data are sensitive to noise, which could degrade their performance in real-world applications where data is often noisy and irregular.

Appendix

9.1 Hyperparameters

- **BERT (Sentiment Classification):**
  - Batch size: 16
  - Learning rate: 2×10^{-5}
  - Epochs: 5
  - Sequence length: 128 tokens
- **GRU (Sentiment Forecasting):**
  - Batch size: 16
  - Hidden size: 32
  - Sequence length: 15
  - Epochs: 50
  - Dropout: 0.3
  - Forecast horizon: 7 days

9.2 Performance Metrics

BERT

Label	Precision	Recall	F1-score	Support
Negative	0.91	0.95	0.93	332
Neutral	0.99	0.94	0.96	506
Positive	0.98	0.99	0.98	1562
Accuracy			0.97	2400
Macro Avg	0.96	0.96	0.96	2400
Weighted Avg	0.97	0.97	0.97	2400

Table 1: Test Classification Report

GRU

Metric	Train Score	Validation Score
MSE	0.0001	0.0001
MAE	0.0084	0.0084
RMSE	0.0103	0.0103

Table 2: GRU Performance Metrics

9.3 Dataset Details

The dataset used in this study consists of 12,000 Uber reviews, each containing metadata such as review content, date, score, and app version. The reviews were preprocessed using custom functions for cleaning, tokenization, and sentiment labeling.

9.4 Figures & Visualizations

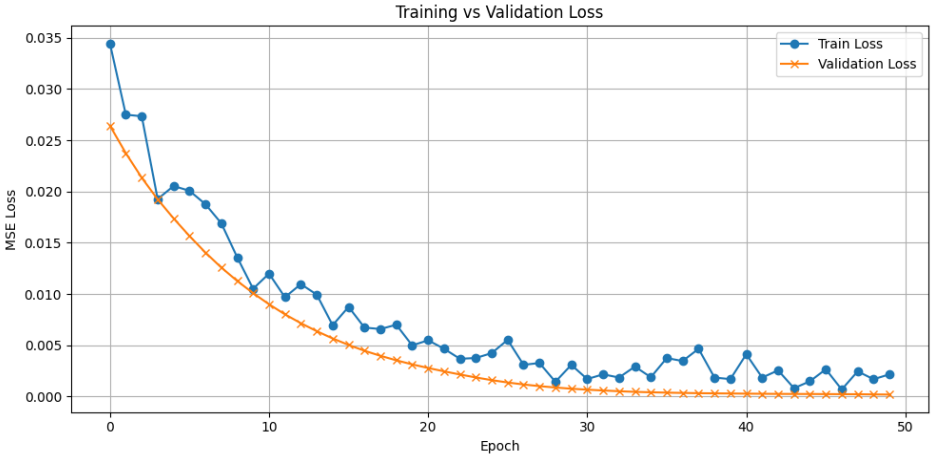

Figure 3: Training and Test Loss Plot. The plot indicates that the model is not overfitting and is generalizing well across the training and test data.

Not Overfitting:

- **Consistent Loss Decrease:** Both training and validation losses decrease steadily.
- **No Divergence:** Training loss and validation loss stay close, showing good generalization.
- **No Saturation:** Loss continues to improve, indicating that the model is still learning.

Not Underfitting:

- **Steady Improvement:** Loss decreases significantly from the beginning, showing that the model is learning.
- **Good Convergence:** No early plateau, indicating that meaningful learning continues.

Conclusion: The model is neither overfit nor underfit, showing a good balance in learning.