Uber Reviews Sentiment Analysis Using Forecasting

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

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

Project Definition

Forecast Sentiment Based on Textual Review Content

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 are sentiment analysis with time series forecasting, by doing so we can find the existing patterns in the data and how the user sentiments evolve. 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.

Abstract

The goal of this research is to Forecast Sentiment Based on Textual Review Content. Objective of this model is to Predict future sentiment trends by analyzing the evolution of textual sentiment in the content column over time. Textual sentiment can reveal nuanced user feelings (e.g., "Very convenient" vs. "Good") that raw scores might miss, helping Uber understand specific pain points or strengths. This paper uses Time series analysis with deep learning methods to predict the sentiment over time.

Literature Survey

Sentiment analysis: Sentiment Analysis: Review of Methods and Models

Sentiment analysis, or opinion mining, is a natural language processing (NLP) subfield that identifies the emotional tone in text, classifying it as positive, negative, or neutral. It has become vital for analyzing user-generated content from social media, e-commerce, and reviews, aiding businesses, researchers, and policymakers in understanding public opinion. This review covers the evolution of sentiment classification methods, from rule-based to deep learning approaches.

1. Rule-Based and Lexicon-Based Methods

Early sentiment analysis used rule-based and lexicon-based methods, relying on predefined word lists to assign sentiment scores (e.g., "happy" as positive, "sad" as negative). Techniques like tokenization and negation handling improve accuracy, with VADER being a key example tailored for social media, including emoticons and slang. These methods are simple but struggle with context, sarcasm, and domain-specific terms.

Copyright © 2025, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

2. Machine Learning-Based Methods

Machine learning (ML) transformed sentiment classification by learning from labeled data. Key algorithms include:

- Naive Bayes: A probabilistic model efficient for short texts like tweets, though less effective with nuanced sentiments.
- Support Vector Machines (SVM): Robust for text classification, especially with TF-IDF, excelling in review analysis.
- Logistic Regression: Simple and interpretable, it performs well on balanced datasets.
- K-Nearest Neighbors (KNN): Effective for small datasets but scales poorly with larger ones.

Feature engineering (e.g., bag-of-words, n-grams) is essential, though ML methods may miss semantic relationships and rely on data quality.

3. Deep Learning-Based Models

Deep learning advanced sentiment classification with neural networks capturing complex patterns. Key models include:

- Recurrent Neural Networks (RNNs): Suitable for sequential text, with LSTMs and GRUs improving long-term dependency capture, ideal for sentence-level analysis.
- Convolutional Neural Networks (CNNs): Adapted from image processing, they efficiently extract features from short texts like posts.
- Transformers and Attention Mechanisms: BERT and similar models use bidirectional context and attention to excel in tasks like aspect-based analysis, outperforming earlier methods on datasets like IMDB reviews.
- Pre-trained Language Models: RoBERTa, GPT, and XL-Net, fine-tuned from large corpora, deliver top performance but demand significant computational power.

Time Series Analysis:

1. Traditional Time Series Models

Traditional models lay the groundwork for forecasting and have been adapted for sentiment data:

- Autoregressive Integrated Moving Average (ARIMA): ARIMA forecasts using past data and errors, applied to Weibo sentiment with LDA clustering to model sentiment shifts, effectively capturing trends.
- Exponential Smoothing (ETS): ETS prioritizes recent data, used with news sentiment to forecast stock prices, showing slight accuracy gains tied to immediate behavioral shifts.
- Seasonal Decomposition: Breaks time series into components, applied to Twitter sentiment to detect recurring opinion patterns, aiding behavioral cycle analysis.

These models are efficient and interpretable but falter with non-linear or complex sentiment-behavior dynamics.

2. Machine Learning-Based Models

Machine learning (ML) improves forecasting by handling non-linearities and multivariate data, ideal for sentiment:

- Support Vector Regression (SVR): SVR enhances stock price predictions using Twitter sentiment, modeling investor behavior.
- Random Forests: Forecasts cryptocurrency prices with social media sentiment, revealing collective emotional impacts on markets.
- Gradient Boosting: XGBoost, applied to e-commerce trends in Marketing 5.0, boosts consumer behavior predictions by combining sentiment and transaction data.

3. Deep Learning-Based Models

Deep learning transforms forecasting for sentiment and behavior with complex dependency modeling:

- Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM): LSTMs with news and Twitter sentiment predict stock volatility (e.g., -0.7 correlation) and cryptocurrency trends, capturing investor behavior.
- Convolutional Neural Networks (CNNs): CNNs with news sentiment improve short-term financial forecasts by detecting behavioral shifts like panic selling.
- Transformers: Outperform LSTMs in cryptocurrency forecasting using tweet sentiment, leveraging attention for volatile behavior prediction; BERT enhances aspect-based analysis.
- Generative Adversarial Networks (GANs): GANs generate synthetic data, showing potential for sentiment evolution modeling, though accuracy gains are dataset-specific.

Gaps in Current Research

Sentiment analysis integrated with time series analysis has advanced forecasting in domains like finance and public opinion, yet significant gaps persist. Current methods struggle with contextual nuances like sarcasm, often relying on static lexicons or models that miss evolving language patterns, while temporal dependency modeling remains limited, with traditional approaches like ARIMA unable to capture non-linear sentiment shifts and even advanced deep learning models lacking intraday granularity. Multimodal integration is underdeveloped, ignoring non-textual cues like images or audio that could enrich time series predictions. Scalability is a challenge, with deep learning's computational demands clashing with real-time needs, and simpler models lacking sophistication.

Links Used for This Research

- A Survey on Sentiment Analysis Methods, Applications, and Challenges - Springer: https://link.springer.com/article/10.1007/s10462-022-10166-2
- Sentiment Analysis Algorithms and Applications: A Survey - ScienceDirect: https://www.sciencedirect.com/science/article/pii/S2090447914000550
- A Comprehensive Survey on Sentiment Analysis Techniques - IJTech: https://ijtech.eng.ui.ac.id/article/view/11111
- A Multi-Method Survey on the Use of Sentiment Analysis in Multivariate Financial Time Series Forecasting - MDPI: https://www.mdpi.com/2076-3417/13/3/1447
- Time Series Forecasting Using Transformers with Sentiment Analysis on Financial Data - Archivo Digital UPM: https://oa.upm.es/78566/

Dataset Selection

Identification:

Uber Customer Reviews Dataset: https://www.kaggle.com/datasets/kanchana1990/uber-customer-reviews-dataset-2024

Contains:

- Username: The reviewer's username is anonymized for privacy.
- Content: Captures the detailed review text where users share their experiences and opinions.
- Score: A numerical rating (1–5) indicating the user's satisfaction level with the product or service.
- ThumbsUpCount: Represents the number of likes or upvotes a review has received, showing how helpful it is to others.
- ReviewCreatedVersion: Specifies the app version when the review was written, helping track feedback across updates.
- At: A timestamp indicating the exact date and time when the review was posted.
- ReplyContent: Contains the developer's response to the review, addressing user concerns or feedback.
- RepliedAt: Records the date and time when the developer replied, useful for analyzing response times.
- AppVersion: The app version string associated with the review, helping track sentiment changes across different releases.

Justification:

This dataset contains over 12,000 customer reviews of the Uber app collected from the Google Play Store. The reviews provide insights into user experiences, including ratings, feedback on services, and developer responses. The data is cleaned and anonymized to ensure privacy compliance and ethical usage. It serves as a valuable resource for sentiment analysis, natural language processing (NLP) with time series.

Preprocessing:

- Text normalization: Convert all text to lowercase for consistency and remove extra spaces, special characters, emojis.
- Feature Extraction: Extract year, month, and day from At to analyze time-based trends.
- Remove Unnecessary Data: Exclude user images as they are not relevant to the analysis.
- Sentiment Analysis: Use NLP tools like VADER or TextBlob to extract sentiment from review text.
- Categorization: Optionally, classify reviews based on app version for deeper insights.

Methodology

The task of this research is to forecast sentiment based on textual reviews and how they change over time. Sentiment analysis is a potent tool with varied applications across industries. It is helpful for social media and brand monitoring, customer support and feedback analysis, market research, etc. By performing sentiment analysis on initial customer feedback, we can identify a new product's target audience or demographics and evaluate the success of a marketing campaign.The reviews of customers often changes based on the service they get. Forecasting the future sentiments could help organizations to give customers more satisfaction.

1.1 Data: The data collected for this task was taken from Keggle. The data was appropriate for this task because it was having mixed customer reviews and also it was having time stamps on hour basis, which is the most suitable to forecast sentiments. The dataset contains reviews, timestamp with date, score of review. The method used to extract sentiment for the task was Vader/ Bert.

1.2 Model: The model used for this was LSTM.LSTM are a form of RNN that can memorize arbitrary-length sequences of input patterns by capturing connections between sequential data types. due to stochastic gradients' failure, RNNs cannot detect long-term dependencies in lengthy sequences. Researchers proposed several novel RNN models, notably LSTM, to address this issue. LSTM networks are extensions of RNNs designed to learn sequential (temporal) data and their long-term connections more precisely than standard RNNs.

Model Design and Architecture: LSTM

2.1. Explanation of Architecture:

Long Short-Term Memory (LSTM) networks are a type of recurrent neural network (RNN) architecture designed to model sequences and capture long-term dependencies. Developed to address the limitations of traditional RNNs, LSTMs are particularly effective in tasks where context over time is crucial, such as language modeling, machine translation, and time series prediction. Below, we delve into the core components and functioning of the LSTM architecture.

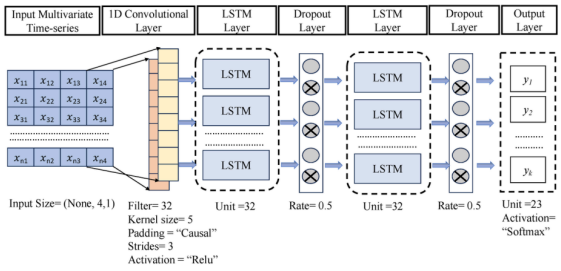

Figure 1: Architecture diagram of a hybrid Conv1D-LSTM model with dropout and softmax output layer.

2.2. Core Components of LSTM

An LSTM network is composed of a series of memory cells, each featuring a unique structure that facilitates the retention and forgetting of information over extended sequences. The cells consist of several key components:

2.2.1. Cell State (Memory Cell) The cell state is a central concept in LSTM networks, acting as a conveyor belt that transports information across the network's sequence of time steps. It allows the network to retain or discard information, thus maintaining long-term dependencies.

2.2.2. Gates Gates are crucial mechanisms within LSTM cells that regulate the flow of information, effectively deciding which information to keep or discard. These gates use sigmoid activations to ensure outputs are between 0 and 1, functioning similarly to binary switches:

- **Forget Gate:** Determines what information should be discarded from the cell state. It takes the previous hidden state and current input to produce an output that scales the cell state. forget gate unit f(t)i (for time step t i and cell), that sets this weight to a value between 0 and 1 via a sigmoid unit:

$$f_i^{(t)} = \sigma\left(b_i^f + \sum_j U_{i,j}^f x_j^{(t)} + \sum_j W_{i,j}^f h_j^{(t-1)}\right), \quad (1)$$

- **Input Gate:** Decides which new information should be stored in the cell state. This involves filtering the input through a sigmoid function and a tanh function to create candidate values to add to the cell state. :

$$s_i^{(t)} = f_i^{(t)} s_i^{(t-1)} + g_i^{(t)} \sigma\left(b_i + \sum_j U_{i,j} x_j^{(t)} + \sum_j W_{i,j} h_j^{(t-1)}\right), \quad (2)$$

- **Output Gate:** Controls what parts of the cell state are output to the next hidden state. This gate influences the next step's input and the final output of the LSTM cell :

$$q_i^{(t)} = \sigma\left(b_i^o + \sum_j U_{i,j}^o x_j^{(t)} + \sum_j W_{i,j}^o h_j^{(t-1)}\right) \quad (3)$$

2.2.3. Hidden State The hidden state is updated at each time step and acts as a short-term memory, capturing information from the current and past inputs. It interacts closely with the gates and cell state to update and carry forward the necessary information.

2.3. Working Mechanism of LSTM

Each LSTM cell processes information in a step-by-step manner: Input Reception: The cell receives the current input and the previous hidden state.

1. **Forget Gate Decision:** Determines which information from the cell state should be forgotten.
2. **Input Gate Decision:** Decides which information will be added to the cell state.
3. **Cell State Update:** Combines the forget and input gate information to update the cell state.
4. **Output Gate Decision:** Determines which part of the updated cell state should be output as the new hidden state.

2.4. Advantages of LSTM

1. **Handling Long Sequences:** LSTMs are adept at managing long sequences due to their ability to retain information over extended periods.
2. **Mitigating Vanishing Gradient Problem:** By using gates, LSTMs reduce the vanishing gradient issue common in traditional RNNs, allowing them to learn long-term dependencies more effectively.
3. **Versatility:** LSTMs are suitable for various applications, including speech recognition, language translation, and more, wherever sequence prediction is needed.

Training Setup/Strategy

The model training followed a structured pipeline including data preprocessing, feature engineering, data scaling, sequence creation, and model training with early stopping to prevent overfitting. The detailed approach is given below:

1. **Feature Engineering:** A 7-day rolling average of the sentiment compound score was created to smooth out short-term fluctuations.
2. **Data Scaling:** StandardScaler was used to normalize the rolling sentiment values, which helps stabilize the learning process.
3. **Data Splitting:** The dataset was split into 80% training and 20% testing sets.
4. **Model Training:** An LSTM-based deep learning model was used, consisting of 4 stacked LSTM layers (each with 100 units) and Dropout layers (0.3) to reduce overfitting. The model used a linear activation in the final Dense layer to predict sentiment scores.

Pseudocode:

Algorithm 1 Train LSTM Model for Sentiment Forecasting

```
Input: CSV file with sentiment scores
Load data from reviews.csv
Compute 7-day rolling mean of compound scores
Fill missing values with mean of rolling scores
Scale the rolling scores using StandardScaler
Create sequences of length 7 for LSTM input
Split sequences into 80% train and 20% test sets
Initialize Sequential LSTM model
for 4 LSTM layers do
    Add LSTM layer with 100 units
    if not last layer then
        Set return_sequences=True
    else
        Set return_sequences=False
    end if
    Add Dropout layer with rate 0.3
end for
Add final Dense output layer with 1 unit (linear activation)
Compile model with Adam optimizer and MSE loss
Setup early stopping (monitor="loss", patience=10)
Train model for 100 epochs, batch size 64
```

Loss function: Mean Squared Error

Mean Squared Error (MSE) is a crucial concept in the field of statistics and machine learning. It serves as a measure to evaluate the accuracy of a model by quantifying the difference between observed and predicted values.Mean Squared Error is a metric used to assess the quality of a predictive model. It is the average of the squares of the errors, where the error is the difference between the actual and predicted values. The formula for MSE is:

$$\text{MSE}_{\text{test}} = \frac{1}{m}\sum_i \left(\hat{\mathbf{y}}_i^{(\text{test})} - \mathbf{y}_i^{(\text{test})}\right)^2 \quad (4)$$

MSE is widely used because it penalizes larger errors more heavily due to the squaring of the differences. This means that it effectively highlights models that make significant prediction errors. Here are some reasons MSE is favoured:**Sensitivity:** MSE is sensitive to large errors, making it useful in applications where bigger mistakes are costly. **Differentiability:** The function is differentiable, which is helpful in optimization and helps in gradient-based learning algorithms. **Interpretability:** Provides a clear measure of model performance, with lower MSE values indicating better model accuracy.

4.2. **Alternatives to Mean Squared Error :-** Depending on the application, other metrics may be more suitable:

* **Mean absolute error (MAE):** Measures the average magnitude of errors, providing a more interpretable metric when units are important.

* **Root Mean Squared Error (RMSE):** Takes the square root of MSE, bringing the error metric back to the same unit as the output variable.
* **R-squared:** Provides a relative measure of how well the model explains the variability of the outcome.