Temporal Analysis of Customer Sentiment Dynamics in Amazon Reviews

Dhruv Jain 202418020
Kaustubh Wade 202418024
Anujit Nair 202418036
Naman Gandhi 202418038

Abstract

Understanding how user sentiment evolves over time in online reviews is critical for businesses. This research analyzes sentiment dynamics in Amazon reviews for Cell Phones and Accessories.The study employs advanced sentiment analysis techniques to explore temporal sentiment patterns, helping businesses proactively manage customer satisfaction, identify emerging trends, and optimize strategic planning.

1 Problem Definition

Understanding evolving customer sentiment is crucial for business success, yet traditional sentiment analysis offers only static snapshots, missing critical trends over time. This limitation prevents businesses from anticipating shifts caused by product updates, promotions, or market changes, leading to missed opportunities and reduced customer satisfaction.

To address this gap, our study analyzes temporal sentiment trends using Amazon reviews data. We develop a pipeline combining text-based sentiment scoring with star ratings, applying time-series forecasting to predict future sentiment patterns. This enables proactive identification of emerging issues and shifting preferences.

By moving from static analysis to dynamic forecasting, businesses gain actionable insights to enhance decision-making, strengthen competitiveness, and improve customer relationships. Our approach transforms review data into a strategic tool for anticipating market changes and optimizing responses.

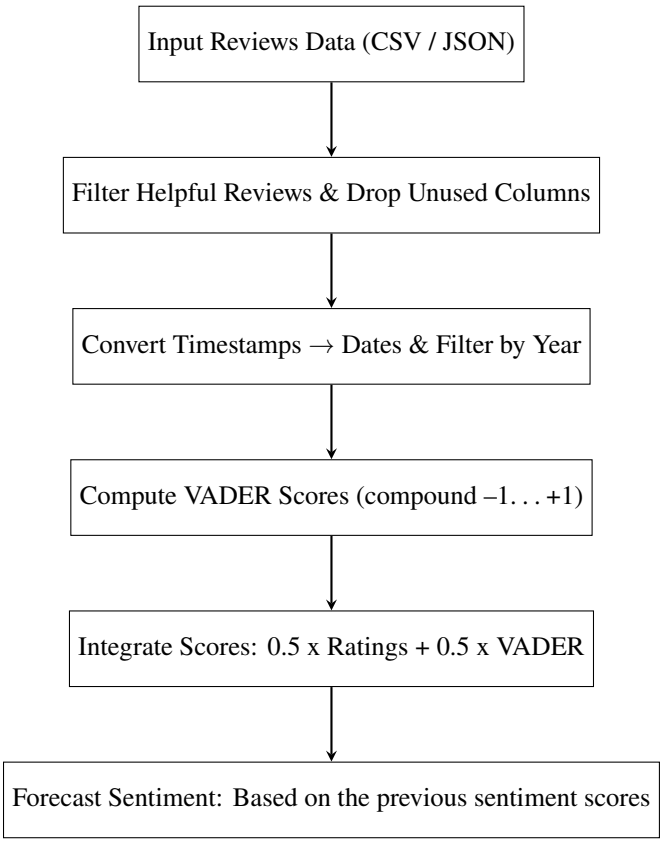

Figure 1: Workflow for preprocessing Amazon review data, extracting and integrating sentiment scores, and classifying overall sentiment.

2 Literature Review

Sentiment analysis has become a vital tool for enhancing time series forecasting in finance, literature, and public opinion research. This review examines key studies integrating sentiment analysis with time series methods, focusing on methodologies, findings, and limitations.

2.1 Evolution of Sentiment Classification Models

Early sentiment analysis relied on supervised models (Naive Bayes, SVM), later shifting to deep learning (RNNs, LSTMs, Transformers). While innovative self-supervised techniques for literary sentiment arcs exist, their application to Amazon reviews is limited due to differences in data structure (brief, noisy reviews vs. coherent narratives). A major flaw is the lack of forecasting capability—the model identifies but does

not predict sentiment shifts. However, its ensemble approach (lexical + Transformer models) could be adapted for reviews if retrained and augmented with forecasting components [1].

2.2 Foundations of Temporal Analysis in Time-Series Data

This study shows sentiment improves multivariate time-series predictions in finance, but its focus on stock prices and Twitter data mismatches product reviews. Sentiment is used as an input feature (to predict prices), whereas our goal is to predict sentiment itself as the target variable [2].

2.3 From Static to Dynamic Sentiment Analysis

A LSTM-BERT hybrid captures temporal sentiment dependencies but is computationally heavy for large-scale reviews. Optimized for sharp financial news fluctuations, it poorly handles gradual sentiment shifts in reviews. A lighter architecture (e.g., DistilBERT) with review-specific preprocessing could be tested but may sacrifice accuracy [3].

2.4 Sentiment Integration in Financial Forecasting

This outdated SVM-based study focuses on cryptocurrency markets, ignoring nuanced sentiment (e.g., feature-specific opinions) crucial for product reviews. Its VECM model's regime-switching logic could theoretically adapt to sentiment phases but requires retooling for review metadata (e.g., star ratings) [4].

2.5 Challenges in Sentiment Regime Detection

Regime detection identifies sentiment phases but struggles with noise and short-term volatility. Current methods lack granularity for mixed sentiments (e.g., 3-star reviews) and mechanisms to handle review-specific challenges (spam, fake reviews). The study correlates sentiment with polls but does not forecast trends [5].

2.6 Detecting and Explaining Causes from Text for Time Series Events

A novel framework links text to time series via Granger causality and a knowledge graph (CGraph). It identifies causal features (N-grams, sentiments) and generates interpretable explanations. While promising, its complexity (FrameNet semantics) may be excessive. A simplified version focusing on verb-noun pairs ("battery drains") could offer actionable insights for business decisions [6].

3 Dataset Selection

3.1 Identification

For this study, we have selected the Amazon Reviews Dataset specifically focusing on the Cell Phones and Accessories category. This dataset comprises comprehensive user-generated reviews, including:

- Textual feedback: Detailed written opinions from customers
- Numerical ratings: Star ratings (1-5 scale) accompanying each review
- Temporal metadata: Precise timestamps for each review submission
- Additional metadata: Helpfulness votes (user-indicated usefulness), ASIN (Amazon Standard Identification Number), and parent ASIN (for product variants)

3.2 Justification

The Amazon Reviews dataset is well suited for analyzing sentiment evolution over time due to the following reasons:

- **Time-stamped Reviews**: The dataset contains timestamps, allowing us to analyze sentiment trends over different periods.
- **Large-scale Data**: The dataset includes a substantial number of reviews, ensuring statistical robustness.
- **User Feedback Insights**: Helpful votes provide a quality measure that allows us to focus on impactful reviews.

3.3 Preprocessing

To prepare the dataset for analysis, we applied the following preprocessing steps:

- **Timestamp Processing**: Converted timestamps into date format for easier time-series analysis.
- **Helpful Reviews Selection**: Included only reviews that were marked as helpful at least once, ensuring relevance.
- **Text Cleaning**: Text normalization was performed for consistency.
- **Sentiment Scoring**: Applied sentiment analysis techniques to assign sentiment scores to each review.

These preprocessing and enhancement steps ensure a high-quality dataset for analyzing sentiment evolution effectively.

4 Methodology

Our project introduces a predictive framework that analyzes the evolution of customer sentiment over time using a LSTM (LSTM) network. The model processes blended sentiment scores—combining textual sentiment from VADER and star ratings—organized into 21-day chronological windows. This approach captures both short-term fluctuations and long-term trends, with data normalized and split into training (80

The LSTM architecture processes sequences bidirectionally, learning patterns that influence sentiment changes in both temporal directions. Hyperparameters like learning rate, batch size, and sequence length are optimized via grid search, while early stopping prevents overfitting. The model minimizes Mean Squared Error (MSE) to generate accurate 3-day sentiment forecasts, validated through low RMSE and MSE on test data.

By transforming static sentiment analysis into dynamic predictions, our framework enables businesses to proactively address emerging trends. Companies can anticipate customer satisfaction shifts, adjust marketing strategies, and allocate resources efficiently. This temporal modeling approach turns customer feedback into a strategic tool for data-driven decision-making in competitive markets.

5 Model design and architecture

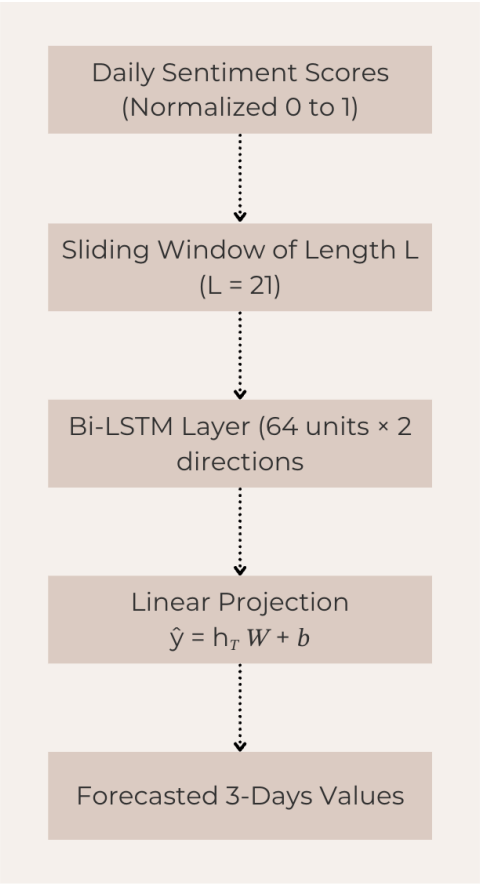

Figure 2: LSTM Forecasting Pipeline: input windows of past sentiment feed a LSTM whose final state is linearly projected to multi-week forecasts.

Model Design

Our forecasting pipeline consists of five sequential components, as illustrated in the flowchart:

1. **Input Normalization**

- **Data**: A univariate time series of weekly sentiment indices (blended star ratings × 0.5 + VADER text scores × 0.5).
- **Normalization**: Scale each week's score into [0, 1].

2. **Sliding-Window Generator** - **Window Length** (L): Group the normalized series into overlapping windows of the past L days, e.g.

$$X^{(i)} = [s_i,\, s_{i+1},\, \ldots,\, s_{i+L-1}], \quad Y^{(i)} = [s_{i+L},\, \ldots,\, s_{i+L+H-1}],$$

where H is the forecast horizon.

3. **LSTM Encoder**

- **Architecture**: A single LSTM with 64 hidden units in the forward pass.

4. **Linear Projection Head**

- **Fully-Connected Layer**: A dense (linear) layer maps $\mathbf{h}_T \in \mathbb{R}^{64}$ to the H-dimensional forecast $\hat{\mathbf{y}} \in \mathbb{R}^H$ via

$$\hat{\mathbf{y}} \;=\; \mathbf{h}_T\, W + \mathbf{b},$$

where $W \in \mathbb{R}^{64\times H}$ and $\mathbf{b} \in \mathbb{R}^H$. 5. **Output Interpretation**

- **Forecast**: $\hat{\mathbf{y}} = [\hat{y}_1, \ldots, \hat{y}_H]$ gives the predicted sentiment for the next H days.

- **Dot Product Detail**: Each forecast step $\hat{y}_h$ is computed by the dot product

$$\hat{y}_h \;=\; \sum_{i=1}^{128} h_{T,i}\, W_{i,h} \;+\; b_h.$$

This design cleanly separates temporal encoding (LSTM) from the regression head, making it easy to adjust window size L, hidden units, or forecast horizon H independently.

Algorithm 1 Temporal Sentiment Forecasting Training Strategy

Initialize Parameters:
  Batch size: $B = 64$
  Sequence length: $L = 21$ days
  Learning rate: $\eta = 0.001$
  Patience: $P = 5$ (early stopping)
  Validation split: 10%
  Test split: 10%
Model Architecture:
  LSTM Layers: 1
  Hidden units: 64 per Direction
  Dense output layer: Linear Activation
Pretraining Phase:
Load the calculated sentiment scores $E \in \mathbb{R}^{|V|\times d}$
for each review $p \in$ reviews **do**
    $X_p, Y_p \leftarrow$ create_sequences(D_p, L)
    $X_p \leftarrow$ normalize(X_p)
    $\theta \leftarrow$ initialize_weights()
end for
Main Training Loop:
while not converged **and** epochs < 50 **do**
    for each batch $(X_b, Y_b) \in$ train_loader **do**
      Forward pass:
      $H \leftarrow \text{LSTM}(E(X_b))$
      $\hat{Y}_b \leftarrow \text{Dense}(H)$
      Loss computation:
      $\mathcal{L}_{\text{MSE}} \leftarrow \frac{1}{B}\sum_{i=1}^{B}(\hat{Y}_b^{(i)} - Y_b^{(i)})^2$
      Backpropagation:
      $\theta \leftarrow \theta - \eta\nabla_\theta(\mathcal{L}_{\text{MSE}})$
    end for
    Validation:
    val_loss $\leftarrow$ evaluate(D_{val})
    if val_loss not improved for P epochs **then**
      break
    end if
end while
Testing & Deployment:
test_metrics $\leftarrow$ evaluate(D_{test})
Save model θ^* with best validation performance
return θ^*, test_metrics

6 Training setup/strategy

We train the LSTM forecaster in a single, end-to-end stage using chronologically ordered sliding windows of past sentiment scores to predict the next block of weeks. First, we

fix random seeds for reproducibility and avoid any shuffling that would break temporal order—every window remains in strict time sequence. We then split our windows into 80% for training, 10% for validation, and 10% for testing, ensuring that the validation and test sets strictly follow the training set in time (no "peeking" into future data).

All input windows are scaled to [0,1] and batched (batch size 64). We feed each batch into a single LSTM layer (64 units). The final hidden state is projected through a linear output layer to produce our multi-days forecast. We optimize with the Adam optimizer (initial learning rate of 0.0001), minimizing Mean Squared Error on the validation set in each epoch.

To prevent overfitting and to adapt learning rates, we employ a callback:
1. **EarlyStopping** halts training when validation loss shows no improvement for 5 epochs, then restores the best weights.

We cap training at 50 epochs but generally observe convergence in 20–30 epochs on GPU. Throughout training we log both training and validation loss curves (MSE) to TensorBoard for real-time monitoring and hyperparameter adjustment. The final model checkpoint is the one with the lowest validation MSE.

7 Loss function

We optimize the **Mean Squared Error (MSE)** between the true future sentiment values and our model's predictions. Formally, if for each sliding window indexed by i our model outputs a vector of H future steps

$$\hat{\mathbf{y}}^{(i)} = \left[\hat{y}_1^{(i)}, \hat{y}_2^{(i)}, \ldots, \hat{y}_H^{(i)}\right]$$

and the ground-truth is

$$\mathbf{y}^{(i)} = \left[y_1^{(i)}, y_2^{(i)}, \ldots, y_H^{(i)}\right],$$

then over N training windows the loss is

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{H}\sum_{h=1}^{H}\left(y_h^{(i)} - \hat{y}_h^{(i)}\right)^2 .]$$

Minimizing this loss penalizes large prediction errors quadratically, encouraging LSTM to produce smooth, trend-aligned forecasts. Additionally, at inference time we often report the **Root Mean Squared Error (RMSE),Mean Absolute Error (MAE)**,

$$\text{RMSE} = \sqrt{\text{MSE}},$$

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{H}\sum_{h=1}^{H}\left(y_h^{(i)} - \hat{y}_h^{(i)}\right).$$

because it rescale-adjusts the error back into the original sentiment units, making performance more interpretable.

8 Experiment Setup

The dataset used in this study was sourced from Amazon's Cell Phones and Accessories product category. It originally existed in JSON format, containing metadata and textual reviews from customers. To facilitate analysis, the dataset was converted into CSV format, filtered, and processed to extract only those reviews that had a helpfulness score greater than zero. This ensured that only meaningful and engaged reviews were retained. Key fields utilized include:

- **text**-Review Body
- **rating** – Integer score (1 to 5).
- **timestamp** – Epoch time indicating the date of submission.

The processed dataset covered a span of multiple months and included tens of thousands of records, sufficient to support time series modeling at a daily granularity.

8.1 Data Processing and Sampling

Text preprocessing was conducted through several NLP techniques:

Sentiment Analysis was carried out using the VADER sentiment analyzer[7]. The compound sentiment score was normalized to a 0–1 range and combined with the normalized user rating (scaled to 0–1) to compute a weighted sentiment score using the formula:

Final Score=(0.5×Rating Score)+(0.5×Sentiment Score)

This resulted in a daily sentiment score series, which was resampled and interpolated to fill missing days.

The time series was then transformed using a sliding window approach:

- Window Length (L): Number of past days used to predict future values.
- Forecast Horizon (H): Number of future days to be predicted.
- Split: 80% training, 20% testing, strictly respecting chronological order.

8.2 Training Details

Each of the deep learning models was trained using consistent hyperparameter settings unless otherwise stated. The following configuration was used:

- Optimizer: Adam (`learning_rate` = 0.001)
- Loss Function: Mean Squared Error (MSE),Mean Absolute Error(MAE)
- Batch Size: 64
- Epochs: 50 (with early stopping enabled on validation loss with a patience of 5 epochs)
- Weight Initialization: Glorot Uniform (default in TensorFlow Keras)
- Callbacks: Early stopping and model checkpointing to prevent overfitting

8.3 Evaluation Metric

The primary evaluation metric used in this study is Mean Squared Error (MSE) and Mean Absolute Error(MAE). MSE measures the average of the squared differences between predicted and actual values. MSE penalizes larger errors more heavily, making it effective for assessing forecasting accuracy where large deviations are undesirable. While RMSE was tracked during training for interpretability, only MSE and MAE was used for model training and final evaluation, aligning the optimization objective with the metric of interest.

9 Implementation of Model

A **Bidirectional Long Short-Term Memory** network was used to capture both past and future temporal dependencies. The architecture consisted of:

- An input layer accepting (L, 1) sequences.
- A LSTM(64) layer.
- A Dense(H) output layer predicting the next H time steps.

The model was compiled with the Mean Squared Error (MSE) loss and the Root Mean Squared Error (RMSE) metric. The dataset was split into training and test sets (80:20 split). The model was trained for up to 50 epochs with a batch size of 32. Validation was monitored for early stopping. After training, predictions were generated and compared to true sentiment scores over time.

Insights:

Mean Squared Error (MSE) on the test data: 0.0001

This low MSE value confirms the model's ability to closely approximate the actual sentiment trajectory.

The LSTM forecast appears smoother than the actual sentiment trajectory, especially during periods of high volatility. While this indicates that the model has learned the general trend, it also suggests a limited sensitivity to sudden fluctuations or local anomalies in the sentiment data. This is typical of many LSTM-based models, which often prioritize overall trend patterns over high-frequency variance.

10 Baseline Model

Moving Average: Averages the last L input points and extends this average across the forecast horizon. This provides a simple statistical grounding for comparison.

MSE: 0.0001

An MSE of 0.0001 reflects numerically decent forecasting accuracy. This confirms that when the underlying sentiment data is relatively smooth or trending, the moving average performs adequately. It also highlights that the sentiment time series may not exhibit excessive volatility over long durations.

The moving average always trails the actual sentiment. Peaks and troughs appear delayed, as past values dominate the averaging window. Sudden upward or downward movements in sentiment are underrepresented. The model smooths transitions too aggressively, missing short-term reactions to real-world events or review shifts.

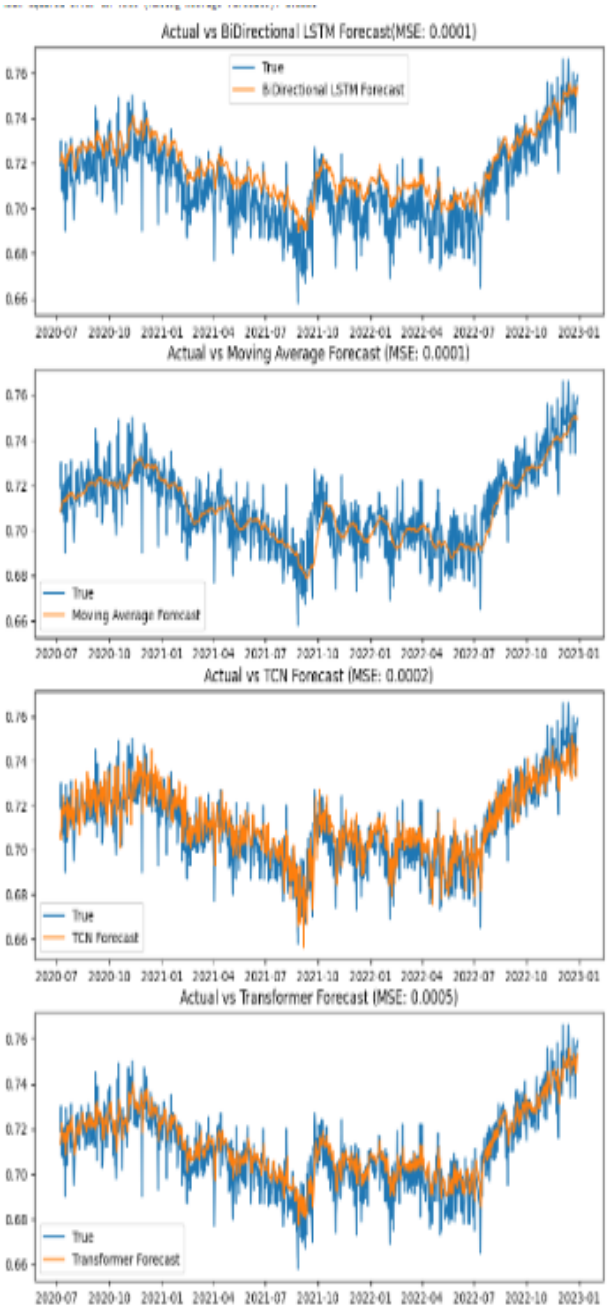

Figure 3: Implementation of All Models

11 SOTA Architectures

1. **Temporal Convolutional Network (TCN)**

   The Temporal Convolutional Network architecture was implemented using the tcn Keras module. Its structure included:

   - Dilated causal convolutions with kernel size 3 and filters = 64.
   - Dilation rates of [1, 2, 4, 8, 16], enabling long receptive fields.
   - Dropout and batch normalization for regularization.

   Unlike LSTMs, TCNs process input sequences in parallel and maintain temporal ordering through causal padding. Their non-recurrent nature offers better gradient propagation and training speed. This model is particularly adept at handling long-range dependencies in sequential data.

   MSE:0.0002

   The TCN (Temporal Convolutional Network) forecast closely aligns with the true values, as indicated by the low Mean Squared Error (MSE) of 0.0002. This suggests the model is highly accurate in predicting the given time series data. The graph likely depicts a time series with consistent trends or seasonal patterns, as the forecast (TCN) closely follows the true values over multiple periods (from July 2020 to January 2023). This indicates the model captures underlying patterns effectively.

2. **Transformer**

   A custom Transformer architecture was constructed to capture global attention patterns in the input sequence. It comprised:

   - Input embedding via a Dense(`d_model`) layer.
   - Positional Encoding layer to preserve sequence order.
   - Multi-Head Self-Attention with `num_heads` = 2 and `key_dim` = `d_model`.
   - Residual connections, Layer Normalization, and a Feed-Forward Network (FFN).

   Transformers are well-suited to capturing long-term dependencies, especially in data where patterns are not strictly local[8]. Their performance was tested across different window sizes to identify optimal configurations.

MSE:0.005

The Transformer forecast shows good alignment with the true values, though the Mean Squared Error (MSE) of 0.005 is higher than the TCN model's MSE (0.0002) from the previous image. This suggests the Transformer is slightly less accurate for this specific dataset but still performs well overall. The Transformer model successfully follows the overall direction of the sentiment series. It captures the gradual decline during 2021 and the recovery phase throughout 2022, indicating that it learns long-term temporal dependencies effectively. The Transformer forecast fails to react to short-term sentiment swings, especially around early 2021 and mid-2022. The orange line lags behind or smooths over several local peaks and valleys, demonstrating a delay in capturing short-term dynamics.

12 Results

Models	MSE	RMSE	MAE
Moving Average Forecast	0.000112	0.010568	0.008267
BiDirectional LSTM	0.000148	0.012182	0.009709
Temporal Convolutional Network	0.000246	0.015670	0.012441
Transformer	0.000543	0.023302	0.018950

Figure 4: Results of Models

The model evaluation was conducted using three key performance metrics: MSE, RMSE, and MAE, across a variety of architectures. Among the evaluated models, the Moving Average Forecast achieved the lowest error across all metrics, indicating that the sentiment trend in the dataset is relatively smooth and predictable without the need for complex architectures. The LSTM model performed best among the deep learning approaches, suggesting its effectiveness in capturing local dependencies and sequential patterns.

Visual plots of actual vs predicted sentiment further reinforce these observations. The Moving Average model closely tracks the true sentiment line, especially during stable and transitional periods. Similarly, the LSTM provides reasonably accurate forecasts with minor deviations during abrupt trend changes. In contrast, the TCN model introduces more fluctuations and fails to smooth transitions effectively, while the Transformer exhibits the most significant divergence, often overfitting local noise and lagging during long-term trend shifts.

Overall, both the numerical metrics and visual evaluations highlight that simple statistical models may outperform deep learning architectures when the input data is stable, univariate, and exhibits strong autocorrelation. However, with more expressive features or multivariate inputs, the performance of deep models could potentially improve.

13 Limitations

Despite the promising setup, several limitations were observed:

1. Model Underfitting: Despite the large dataset, deep models may have underfit due to simplistic or overly smoothed input signals (e.g., daily average sentiment).

2. Feature Limitation: The input was limited to univariate sentiment scores; rich contextual features were not incorporated, which may limit expressiveness.

3. Architectural Mismatch: Some architectures like Transformers may be less suited to low-variance, univariate time series without added complexity (e.g., categorical embeddings or positional context).

4. Limited Tuning: Advanced models like Transformers and TCNs require more sophisticated hyperparameter tuning and architectural experimentation to reach optimal performance.

14 Future Scope

There are several opportunities to enhance the work:

1. Real-Time Forecasting Deployment Extend the model into a real-time prediction system where new reviews are ingested continuously, and sentiment forecasts are updated dynamically. This would have practical applications in reputation monitoring for e-commerce platforms.

2. Model Interpretability Explainability Integrate explainability tools (e.g., SHAP, LIME, or attention visualizations) to understand what influences sentiment trends and model predictions. This can be critical for stakeholders in marketing and customer experience.

3. Cross-Product or Multi-Domain Forecasting Generalize the pipeline to work across multiple product categories, enabling scalable sentiment trend analysis across sectors like electronics, fashion, or books.

4. Sentiment Classification Fusion Combine time series forecasting with classification tasks (e.g., detecting sentiment spikes or anomalies), turning the model into a hybrid predictive and diagnostic tool.

5. Robustness to Data Drift Implement drift detection and model adaptation techniques to maintain performance as customer sentiment patterns evolve over time due to seasonality, events, or market shifts.

15 Conclusion

This project aimed to forecast sentiment trends over time using customer review data, bridging natural language processing and time series modeling. By converting textual reviews into a daily sentiment time series using the VADER sentiment analyzer, the work laid a solid foundation for temporal sentiment analysis.

A range of models—from simple baselines like Moving Average to deep learning architectures such as LSTM, Temporal Convolutional Networks (TCN), and Transformers—were implemented and evaluated. Surprisingly, the Moving Average baseline outperformed the more complex models in terms of RMSE and MAE, highlighting that sentiment trends in the dataset are relatively smooth and may not benefit significantly from high-capacity models without additional features or tuning.

The LSTM model showed promising results and was the most effective among the deep learning approaches, reinforcing its strength in modeling sequential dependencies in univariate time series. However, the performance of Transformer and TCN models suggests that further tuning or richer input representations may be necessary to leverage their full potential.

Overall, the project demonstrates the feasibility of forecasting sentiment trends from textual data and offers insights into the suitability of different modeling techniques. It opens avenues for deploying sentiment forecasting tools in real-time systems, enriching feature inputs, and adopting newer state-of-the-art architectures for further performance gains.

16 References

1. Chun, Jon. "SentimentArcs: A novel method for self-supervised sentiment analysis of time series shows SOTA transformers can struggle finding narrative arcs." arXiv preprint arXiv:2110.09454 (2021).

2. Liapis CM, Karanikola A, Kotsiantis S. A Multi-Method Survey on the Use of Sentiment Analysis in Multivariate Financial Time Series Forecasting. Entropy. 2021; 23(12):1603.

3. Chou, Hsiao-Chuan, "Combination of Time Series Analysis and Sentiment Analysis for Stock Market Forecasting" (2021). USF Tampa Graduate Theses and Dissertations.

4. Georgoula, Ifigeneia and Pournarakis, Demitrios and Bilanakos, Christos and Sotiropoulos, Dionisios and Sotiropoulos, Dionisios and Giaglis, George M., Using Time-Series and Sentiment Analysis to Detect the Determinants of Bitcoin Prices (May 17, 2015).

5. O'Connor, B., Balasubramanyan, R., Routledge, B. and Smith, N. 2010. From Tweets to Polls: Linking Text Sentiment to Public Opinion Time Series. Proceedings of the International AAAI Conference on Web and Social Media. 4, 1 (May 2010), 122-129. DOI:https://doi.org/10.1609/icwsm.v4i1.14031

6. Kang, Dongyeop, et al. "Detecting and explaining causes from text for a time series event." arXiv preprint arXiv:1707.08852 (2017).

7. C. Hutto and E. Gilbert, "VADER: A Parsimonious Rule-Based Model for Sentiment Analysis of Social Media Text", ICWSM, vol. 8, no. 1, pp. 216-225, May 2014.

8. Lim, B., Arık, S. Ö., Loeff, N., Pfister, T. (2021). Temporal fusion transformers for interpretable multi-horizon time series forecasting. International Journal of Forecasting, 37(4), 1748–1764.