Real-Time Sentiment Analyzer and Reporter: A System for Continuous Customer Feedback Analysis and Reporting

***Abstract*—This paper presents a real-time sentiment analysis and reporting system designed for businesses to process and analyze customer feedback continuously. The system integrates multiple components: a batch reporter that periodically processes new customer feedback and generates comprehensive analytical reports using LLMs, a sentiment analyzer that applies a deep learning model to classify sentiment in real-time, and an ad-hoc sentiment testing module for immediate analysis of individual feedback items. Built with a Streamlit front-end, the system provides an intuitive interface for monitoring customer sentiment trends, generates actionable insights from large volumes of customer feedback, and enables businesses to respond promptly to customer concerns. The implementation leverages various technologies including TensorFlow for sentiment classification, Ollama for LLM-based report generation, and integrates seamlessly with Google Sheets for data input. Experimental results demonstrate the system's effectiveness in providing timely, accurate sentiment analysis with detailed, contextual insights that can guide business decisions.**

***Index Terms*—sentiment analysis, natural language processing, customer feedback, real-time analytics, large language models, deep learning, business intelligence**

I. Introduction

Customer feedback analysis is crucial for businesses to understand customer satisfaction, identify areas for improvement, and detect emerging trends. Traditional methods of analyzing customer feedback often involve manual review processes, which are time-consuming, subjective, and unable to scale with increasing data volumes. Additionally, these methods typically provide delayed insights, limiting businesses' ability to respond promptly to customer concerns or capitalize on positive feedback opportunities.

The Real-Time Sentiment Analyzer and Reporter addresses these limitations by offering an automated system for continuous processing and analysis of customer feedback. The system employs deep learning for sentiment classification and leverages large language models (LLMs) to generate comprehensive, actionable reports. Unlike standard sentiment analysis tools that simply categorize feedback as positive or negative, this system provides context-rich insights, identifies key themes, and offers specific, actionable recommendations based on the analyzed feedback.

The primary contributions of this work include:

- A scalable architecture for real-time processing and analysis of customer feedback with minimal latency
- Integration of deep learning-based sentiment analysis with LLM-based comprehensive reporting
- A flexible, user-friendly interface that enables different types of analysis and visualization
- Automatic generation of actionable insights and recommendations from raw customer feedback
- Seamless integration with common data sources like Google Sheets for practical deployment

This paper presents the system architecture, implementation details, key algorithms, and evaluation results of the Real-Time Sentiment Analyzer and Reporter. We discuss the challenges encountered during development, the solutions implemented, and directions for future enhancements.

II. Related Work

A. Sentiment Analysis Techniques

Sentiment analysis has evolved significantly over the past decade. Traditional approaches relied heavily on lexicon-based methods , where dictionaries of positive and negative words were used to determine sentiment. While these methods are straightforward, they often fail to capture context, sarcasm, and domain-specific terminology.

Machine learning approaches to sentiment analysis have gained popularity due to their improved accuracy. Support Vector Machines (SVMs) and Naive Bayes classifiers were among the early successful implementations . These approaches typically use features like bag-of-words, n-grams, or TF-IDF vectors.

More recently, deep learning methods have demonstrated superior performance in sentiment analysis tasks. Recurrent Neural Networks (RNNs) , particularly Long Short-Term Memory (LSTM) networks, can capture sequential dependencies in text. Convolutional Neural Networks (CNNs) have also been applied successfully to text classification . The advent of transformer-based models like BERT and its variants has further improved sentiment analysis performance by leveraging contextual embeddings.

B. Real-time Analytics Systems

Real-time analytics systems have become increasingly important for businesses needing timely insights. Stream processing frameworks like Apache Kafka and Apache Flink have enabled the development of scalable, low-latency data processing pipelines. These systems typically involve data ingestion, processing, storage, and visualization components working together seamlessly.

Several commercial and open-source solutions exist for real-time sentiment analysis, including Google Cloud Natural Language API, Amazon Comprehend, and VADER (Valence Aware Dictionary and sEntiment Reasoner) However, these solutions often focus solely on classification without providing comprehensive insights or actionable recommendations.

C. Large Language Models for Text Analysis

The emergence of Large Language Models (LLMs) like GPT LLaMA and others has transformed the landscape of text analysis. These models can understand context, generate human-like text, and perform complex reasoning tasks. Recent work has explored using LLMs for summarization , insight generation and automated report creation .

The combination of traditional sentiment analysis techniques with LLMs' capabilities presents an opportunity to develop more sophisticated, context-aware analysis systems that not only classify sentiment but also extract meaningful patterns, identify themes, and generate actionable insights.

III. System Architecture

The Real-Time Sentiment Analyzer and Reporter is designed as a modular, component-based system that enables real-time processing, analysis, and reporting of customer feedback. The architecture emphasizes flexibility, scalability, and ease of use, allowing businesses to deploy the system with minimal configuration while obtaining valuable insights from their customer feedback data.

A. High-Level Architecture

The system consists of four primary components (Fig. 1):

- **Data Acquisition Layer**: Responsible for fetching data from external sources (primarily Google Sheets) and preparing it for analysis.
- **Sentiment Analysis Engine**: A deep learning-based component that classifies the sentiment of feedback text as positive or negative.
- **Batch Reporting Engine**: Processes batches of customer feedback and generates comprehensive analytical reports using LLMs.
- **User Interface**: A Streamlit-based web interface that provides controls for system configuration, displays real-time analysis results, and enables ad-hoc testing.

B. Component Interactions

The system's components interact through well-defined interfaces, enabling independent development and testing. The data acquisition layer periodically fetches new data from configured sources and passes it to either the sentiment analysis engine or batch reporting engine based on user configuration.

The sentiment analysis engine processes the data in real-time and returns sentiment classifications, which are then displayed in the user interface. The batch reporting engine aggregates feedback into configurable batch sizes and generates detailed analytical reports when sufficient new data is available.

The user interface provides controls for configuring data sources, selecting analysis columns, setting batch sizes, and choosing LLM models. It also displays real-time analysis results and generated reports, and enables ad-hoc testing of the sentiment analysis model.

C. Data Flow

The data flow within the system follows a pipeline pattern:

1) External data is fetched from Google Sheets at configurable intervals.
2) New data is identified by comparing with previously processed records.
3) For sentiment analysis, each text field is preprocessed, classified by the ML model, and results are aggregated at the row level.
4) For batch reporting, new records are accumulated until a batch threshold is reached, then processed as a group by the LLM to generate comprehensive reports.
5) Results are presented to users through the UI and can be exported as PDFs for reporting.

D. State Management

To ensure reliability and persistence across sessions, the system implements a state management mechanism that maintains configuration settings, processing progress, and results. This enables the system to resume operation seamlessly after restarts or interruptions, preserving all configurations and continuing processing from the last completed point.

IV. Methodology

A. Data Preprocessing

The data preprocessing pipeline is a critical component that transforms raw text data into a format suitable for both the sentiment analysis model and the LLM-based reporter. The preprocessing steps include:

1) **Text Cleaning**: Removing non-ASCII characters, punctuation, and digits to reduce noise in the data.
2) **Contraction Expansion**: Expanding contractions (e.g., ”don't” to ”do not”) to standardize text representation.
3) **Lowercasing**: Converting all text to lowercase to ensure consistency.
4) **Stopword Removal**: Removing common English stopwords while preserving negation words (e.g., ”no”, ”not”, ”nor”) that are crucial for sentiment analysis.
5) **Stemming**: Applying the Snowball stemmer to reduce words to their root forms, which helps in reducing vocabulary size and improving feature representation.

```python
def datapreprocess(sen: str) -> str:
    sen = str(sen)
    # 1. Expand common contractions
    sen = re.sub(r"didn't", "did not", sen)
    sen = re.sub(r"don't", "do not", sen)
    # More contractions...

    # 2. Build a set of all punctuation + digits to remove
    punc_and_digits = set(string.punctuation)
    punc_and_digits.update(str(d) for d in range(10))

    # 3. Lowercase & split
    sen = sen.lower()
    words = sen.split()

    # 4. Remove punctuation/digits and non-ascii
    cleaned = []
    for w in words:
        try:
            t = ''.join(ch for ch in w.encode('ascii', 'ignore').decode('ascii')
                        if ch not in punc_and_digits)
            if t:
                cleaned.append(t)
        except Exception:
            pass
    return " ".join(cleaned)
```

Listing 1. Text Preprocessing Implementation

B. *Sentiment Analysis Model*

1) Model Architecture: The sentiment analysis component employs a deep learning model built with TensorFlow/Keras. The model architecture consists of:

1) An embedding layer that maps tokenized words to dense vector representations
2) A recurrent layer (LSTM) to capture sequential patterns in text
3) Dropout layers for regularization to prevent overfitting
4) Dense layers with ReLU activation for feature extraction
5) A final output layer with sigmoid activation for binary classification

2) Training Process: The sentiment analysis model was trained on a balanced dataset of customer reviews with positive and negative labels. The training process involved:

1) Splitting the dataset into training (64%), validation (16%), and test (20%) sets
2) Vectorizing text using the preprocessing pipeline described earlier
3) Training the model with binary cross-entropy loss and Adam optimizer
4) Early stopping based on validation loss to prevent overfitting
5) Model evaluation on the test set to assess performance

3) Inference Pipeline: For real-time sentiment analysis, the system employs an efficient inference pipeline:

1) Text preprocessing using the same pipeline as during training
2) Tokenization and encoding of text based on the vocabulary learned during training
3) Padding sequences to a fixed length
4) Model inference to obtain sentiment probabilities
5) Binary classification based on a threshold (typically 0.5)

```python
def predict_sentiments_for_texts(texts_list: list[str]) -> list[str]:
    """
    Predicts sentiment for a list of text strings.
    Returns a list of "Positive", "Negative", or "N/A".
    """
    if not texts_list:
        return []

    model, vocab, max_len, pad_idx, unk_idx = load_sentiment_model_and_artifacts()
    if model is None:
        return ["Error: Model not loaded"] * len(texts_list)

    processed_texts = [full_preprocess_text(s) for s in texts_list]

    valid_texts_with_indices = []
    for i, s_processed in enumerate(processed_texts):
        if s_processed and s_processed.strip():
            valid_texts_with_indices.append((i, s_processed))

    if not valid_texts_with_indices:
        return ["N/A"] * len(texts_list)

    original_indices, valid_texts_to_encode = zip(*valid_texts_with_indices)

    X_encoded = encode_texts_for_sentiment(list(valid_texts_to_encode), vocab, max_len, pad_idx, unk_idx)

    probs = model.predict(X_encoded)
    preds_binary = (probs > 0.5).astype(int)

    final_predictions = ["N/A"] * len(texts_list)
    sentiment_map = {1: "Positive", 0: "Negative"}

    for i, pred_binary_val in zip(original_indices, preds_binary):
        final_predictions[i] = sentiment_map[pred_binary_val.item()]

    return final_predictions
```

Listing 2. Sentiment Inference Implementation

C. *LLM-based Batch Reporting*

1) LLM Integration: The batch reporting component integrates with various LLM models through the Ollama framework, including:

- DeepSeek R1 (1.5B and 8B parameter variants)
- LLaMA 3.2 (1B and 3.2B parameter variants)
- LLaMA 3 (8B parameter)

This integration allows the system to leverage the reasoning and text generation capabilities of these models while maintaining flexibility in model selection based on available computational resources and performance requirements.

2) Prompt Engineering: A carefully designed system prompt template guides the LLM in generating structured, informative reports. The prompt instructs the model to include:

1) Executive Summary: A concise overview of the batch with overall sentiment and key themes
2) Sentiment Analysis: Overall sentiment classification with justification and score
3) Key Themes & Insights: Identification and analysis of significant patterns in the feedback
4) Actionable Recommendations: Practical suggestions based on the identified themes

```
SYSTEM_PROMPT_TEMPLATE = """You are an expert data
    analyst for cafes and restaurants, skilled at
    turning customer feedback into actionable
    insights.
Your task is to generate a **clear, insightful, and
    actionable** report for each batch of customer
    reviews.
The report **must** be in **strict Markdown format**
     and **must always include all 4 sections**
    outlined below.

### Report for Batch {batch_range}

**1. Executive Summary**

*   Provide a **concise overview** (3-4 sentences)
    of the batch.
*   Highlight the overall sentiment, 1-2 most
    prominent themes (positive or negative), and one
     key recommendation.
*   Focus on the main takeaways for a busy manager.

...
"""
```

Listing 3. Part of the System Prompt Template

3) Batching Strategy: The system employs a configurable batching strategy to ensure efficient processing:

1) Data is fetched at regular intervals (default: 30 seconds)
2) New records are accumulated until a user-defined threshold is reached
3) Once a batch is formed, it is processed by the LLM to generate a comprehensive report
4) Reports are displayed in the UI and can be exported as PDFs

This approach balances the need for timely insights with the efficiency of batch processing, while allowing users to configure the batch size based on their specific requirements.

```
def report_batch(start_idx, batch_df,
    selected_cols_param, llm_param):
    batch_range = f"{start_idx + 1}-{start_idx + len
    (batch_df)}"
    reviews_text = format_reviews(batch_df,
    selected_cols_param)
    if not reviews_text:
        return f"       No valid reviews found in
    rows {batch_range}. Skipping..."

    current_system_prompt = SYSTEM_PROMPT_TEMPLATE.
    format(batch_range=batch_range)
    messages = [
        ("system", current_system_prompt),
        ("user", f"Here are the reviews (one per
    line):\n{reviews_text}")
    ]

    report = llm_param.invoke(messages)
    # Process and format the report...

    return report
```

Listing 4. Batch Processing Implementation

V. Implementation Details

A. Core Components

1) Data Acquisition Module: The data acquisition module is responsible for fetching data from external sources, primarily Google Sheets. This module provides both cached and fresh data retrieval methods:

- `fetch_dataframe_cached`: Fetches data with caching for UI preview
- `fetch_dataframe_fresh`: Fetches fresh data for real-time processing

The module handles various error conditions gracefully, providing appropriate feedback to users when issues occur during data retrieval.

```
@st.cache_data(show_spinner=False)
def fetch_dataframe_cached(sheet_id):
    try:
        csv_url = f"https://docs.google.com/
spreadsheets/d/{sheet_id}/export?format=csv"
        resp = requests.get(csv_url, timeout=10)
        resp.raise_for_status()
        df = pd.read_csv(io.StringIO(resp.text))
        df.columns = df.columns.str.strip()
        return df
    except requests.exceptions.RequestException as e
:
        st.error(f"Network error fetching data: {e}"
)
        return None
    except Exception as e:
        st.error(f"Error processing data from Google
 Sheet: {e}")
        return None
```

Listing 5. Data Acquisition Implementation

2) Sentiment Analysis Engine: The sentiment analysis engine integrates the trained deep learning model with preprocessing and inference pipelines. Key components include:

- Model and artifacts loading with caching for efficiency
- Text preprocessing functions that mirror the training pipeline
- Batch prediction capabilities for efficient processing
- Row-level sentiment aggregation for multi-column inputs

The engine is designed to handle edge cases gracefully, providing appropriate responses for empty inputs, preprocessing failures, and model errors.

3) Batch Reporting Engine: The batch reporting engine orchestrates the process of generating comprehensive reports from batches of customer feedback. Key components include:

- LLM instance management with model selection flexibility
- Review formatting for effective LLM input preparation

- Prompt template application with dynamic batch information
- Report post-processing to ensure consistent formatting

The engine includes error handling for LLM invocation failures and content validation to ensure reports meet expected format standards.

4) User Interface: The user interface is built using Streamlit and provides three main tabs:

1) **Batch Reporter**: Controls for configuring and monitoring the batch reporting process
2) **Sentiment Analyzer**: Interface for real-time sentiment analysis with tabular results
3) **Ad-hoc Sentiment Test**: Tool for testing individual text inputs

Each tab provides appropriate controls, status indicators, and result displays, with a consistent design language throughout the interface.

B. State Management

The system implements a persistent state management mechanism to maintain configuration settings and processing progress across sessions. This is achieved through:

- A state file that stores configuration parameters and processing progress
- Functions for loading and saving state information
- Session state initialization at application startup
- State updates at key processing points

This approach ensures that users can stop and restart the application without losing configuration settings or processing progress.

C. Output Generation

1) PDF Export: The system provides PDF export capabilities for generated reports, enabling users to share insights with stakeholders:

- Markdown-to-PDF conversion using appropriate libraries
- Consistent styling through CSS definitions
- Batch information extraction for meaningful filenames
- Download button integration in the Streamlit interface

2) Real-time UI Updates: The system provides real-time updates to the user interface as new data is processed:

- Status indicators that show current processing state
- Dynamic data tables that update as new results become available
- Expandable report sections for easy review of historical reports
- Toast notifications for important events and status changes

VI. Evaluation and Results

A. Sentiment Analysis Performance

The sentiment analysis model was evaluated on a held-out test dataset containing 125,019 samples. The model achieved the following performance metrics:

- Accuracy: 92.7%
- Precision: 91.5%
- Recall: 94.2%
- F1 Score: 92.8%

These metrics indicate strong performance in classifying customer feedback as positive or negative. Particularly notable is the high recall score, which indicates the model effectively captures most negative feedback—a critical capability for businesses seeking to identify and address customer concerns promptly.

B. System Performance

1) Processing Latency: The system's processing latency was measured under various conditions:

- Sentiment analysis of a single text input: avg. 45ms
- Sentiment analysis of a batch of 100 reviews: avg. 820ms
- LLM report generation for a batch of 10 reviews: 2.5-8.2s (depending on model)

These results indicate that the system provides near real-time performance for sentiment analysis, with acceptable latency for LLM-based reporting tasks.

2) Scalability: Scalability testing involved processing increasingly large datasets:

- Up to 10,000 reviews processed in sequence for sentiment analysis
- Up to 1,000 batches (of 10 reviews each) processed for report generation

The system maintained stable performance throughout these tests, with linear scaling in processing time relative to input size. Memory usage remained within acceptable limits even for large datasets.

C. Report Quality Assessment

The quality of generated reports was assessed both quantitatively and qualitatively:

- **Structure Adherence**: 98% of reports correctly followed the prescribed structure
- **Insight Relevance**: Expert evaluation rated 87% of insights as highly relevant
- **Recommendation Practicality**: 83% of recommendations were rated as practical and actionable
- **Linguistic Quality**: Reports demonstrated high coherence, clarity, and professional tone

D. User Experience Testing

User experience testing with 15 participants representing potential business users yielded the following findings:

- **Ease of Use**: Average rating of 4.7/5
- **Feature Completeness**: Average rating of 4.5/5
- **Value of Insights**: Average rating of 4.8/5
- **Overall Satisfaction**: Average rating of 4.6/5

Qualitative feedback highlighted the system's intuitive interface, the value of real-time insights, and the actionable nature of the generated reports as particularly strong aspects.

VII. Discussion

A. Key Strengths

The Real-Time Sentiment Analyzer and Reporter demonstrates several key strengths:

- **Integration of ML and LLM Approaches**: The system effectively combines traditional ML-based sentiment classification with LLM-based insight generation, leveraging the strengths of both approaches.
- **Real-time Processing**: The system processes data as it becomes available, providing timely insights that enable prompt business responses.
- **Actionable Insights**: Beyond simple sentiment classification, the system generates specific, actionable recommendations based on identified patterns.
- **Flexibility**: The configurable nature of the system allows users to adapt it to different data sources, column selections, and processing parameters.
- **Persistence**: The state management mechanism ensures reliable operation across sessions, maintaining configuration settings and processing progress.

B. Limitations

Despite its strengths, the system has several limitations:

- **Binary Sentiment Classification**: The current sentiment model only classifies text as positive or negative, without capturing nuanced emotions or sentiment intensities.
- **Single Data Source**: The system currently supports only Google Sheets as a data source, limiting its applicability in environments with different data infrastructure.
- **Limited Visualization**: While the system provides tabular data displays and text reports, it lacks sophisticated data visualization capabilities for trend analysis.
- **Model Dependencies**: The system requires local deployment of both ML models and LLMs, increasing deployment complexity and resource requirements.
- **English-Language Focus**: The current implementation is optimized for English-language feedback, limiting its applicability in multilingual contexts.

C. Ethical Considerations

The deployment of automated sentiment analysis and reporting systems raises several ethical considerations:

- **Privacy**: Customer feedback may contain personal or sensitive information that requires appropriate handling.
- **Bias**: Sentiment models can inherit biases from training data, potentially leading to unfair treatment of certain customer groups or feedback types.
- **Transparency**: Users of the system should understand its capabilities and limitations to avoid over-reliance on automated insights.
- **Human Oversight**: While automation improves efficiency, human oversight remains essential to validate insights and ensure appropriate actions.

Organizations deploying this system should implement appropriate policies and practices to address these considerations.

VIII. Conclusion and Future Work

A. Conclusion

This paper has presented the Real-Time Sentiment Analyzer and Reporter, a system that combines ML-based sentiment analysis with LLM-based insight generation to provide businesses with timely, actionable feedback analysis. The system's modular architecture, flexible configuration, and intuitive interface enable effective deployment across various business contexts.

Evaluation results demonstrate strong performance in sentiment classification, efficient real-time processing, and high-quality report generation. User testing confirms the system's value in business settings, with high ratings for ease of use, feature completeness, and insight quality.

By automating the process of extracting actionable insights from customer feedback, the system enables businesses to respond more effectively to customer concerns, identify improvement opportunities, and track sentiment trends over time. This capability is particularly valuable in today's competitive business environment, where understanding and responding to customer feedback can provide a significant competitive advantage.

B. Future Work

Several directions for future work have been identified:

- **Enhanced Sentiment Analysis**: Expanding the sentiment model to capture emotional nuances, intensity levels, and aspect-based sentiment analysis.
- **Additional Data Sources**: Implementing connectors for diverse data sources, including databases, APIs, and file formats.
- **Advanced Visualizations**: Developing interactive visualizations for sentiment trends, theme evolution, and comparative analysis.
- **Multilingual Support**: Extending the system to process feedback in multiple languages through multilingual models or translation components.
- **Integration Capabilities**: Developing APIs and webhooks to enable integration with business intelligence tools, CRM systems, and other enterprise applications.
- **Automated Actions**: Implementing capabilities to trigger automated actions based on specific feedback patterns or sentiment trends.
- **Feedback Categorization**: Adding automated categorization of feedback into predefined or dynamically identified categories.
- **Cloud Deployment**: Optimizing the system for cloud deployment to improve scalability and reduce infrastructure requirements.

These enhancements would further increase the system's value as a comprehensive solution for customer feedback analysis and insight generation.