Original Article
ISSN (Online): 2350-0530
ISSN (Print): 2394-3629

A LIGHTWEIGHT LSTM FRAMEWORK FOR CONTEXTUAL SENTIMENT CLASSIFICATION

Nidhi [1], Onkar Singh [1], Prince [1], Dr. Monika Garg [1]

[1] Department of Computer Science and Engineering, Echelon Institute of Technology, Faridabad, India

Received 25 May 2024
Accepted 12 June 2024
Published 30 June 2024

DOI
10.29121/granthaalayah.v12.i6.2024.6095

Funding: This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

ABSTRACT

With the rapid increase in opinion-rich content shared across the internet, text sentiment analysis has emerged as a vital tool in both academic research and industrial applications. Sentiment analysis typically involves classifying a piece of text as expressing positive, negative, or neutral emotion. Traditional approaches to text classification often require extensive feature engineering and rely heavily on tokenization and embedding techniques, making them resource-intensive and less adaptive to context. To address these limitations, Long Short-Term Memory (LSTM) networks—an advanced form of Recurrent Neural Networks (RNNs)—have been adopted for their ability to capture long-range dependencies in textual data. This study proposes a sentiment classification model based solely on LSTM architecture to analyze short texts and effectively extract context-aware sentiment patterns. Unlike conventional models, LSTM-based frameworks can learn temporal word relationships without explicit syntactic parsing or handcrafted features. By leveraging the memory capabilities of LSTM, the proposed model enhances sentiment categorization accuracy while maintaining a relatively lightweight computational profile. Experimental evaluations demonstrate the effectiveness of LSTM in capturing contextual semantics, making it a suitable choice for real-time sentiment detection tasks in dynamic and user-generated content environments.

1. INTRODUCTION

In today's data-driven world, the volume of digital content is growing at an exponential pace. With this surge in online content, especially in the form of user-generated opinionated texts, accessing relevant information has become increasingly challenging. Text classification offers a promising solution to this problem by enabling the automatic identification and categorization of sentiments embedded in textual data. The popularity of sentiment analysis has witnessed a sharp rise not only in academic research but also across commercial platforms like Amazon, Flipkart, Myntra, Ajio, and JioMart, where understanding customer feedback in real time holds significant value [1].

Sentiment classification, a subset of text classification, involves determining whether a given text is positive, negative, or neutral. Traditional approaches to sentiment analysis rely on assigning weighted sentiment scores to phrases, identifying the orientation of opinions toward specific objects, trends, or categories. These systems leverage Natural Language Processing (NLP) and Machine Learning (ML) to understand and quantify textual emotions, helping businesses and researchers make informed decisions [2].

1.1. RULE-BASED METHODS

Rule-based sentiment classification methods use manually crafted rules for identifying sentiment polarity. These rules can be based on lexical patterns, keyword matches, or syntactic structures. While these methods are intuitive and easy to implement, they lack the scalability and adaptability required for large-scale applications. Moreover, maintaining and updating these rule sets becomes cumbersome as data grows in complexity and volume [3].

1.2. MACHINE LEARNING-BASED METHODS

Machine Learning-based methods provide a scalable and data-driven approach to sentiment classification. These models learn from labeled training data and can generalize to unseen texts. NLP technologies underpinning these methods include syntactic parsing, tokenization, part-of-speech tagging, and semantic analysis. Techniques such as Logistic Regression, Support Vector Machines, and more recently, Deep Learning models like LSTM (Long Short-Term Memory) have shown considerable promise in handling sequential data and capturing contextual dependencies [4].

LSTM networks, a type of Recurrent Neural Network (RNN), are particularly well-suited for text analysis because they can retain long-term dependencies, making them more effective in understanding the context of a sentence. In contrast to conventional RNNs, LSTMs are designed to combat the vanishing gradient problem and maintain memory across time steps, which is crucial for sentiment tasks that rely on the semantic composition of words and phrases [5].

1.3. PREPROCESSING IN TEXT CLASSIFICATION

Effective preprocessing is crucial for transforming raw textual data into a form amenable to machine learning. This stage typically includes tokenization, which breaks text into words or phrases; part-of-speech tagging, which labels words with grammatical tags; and stemming or lemmatization, which reduces words to their base forms. Removing stop words—common words that carry little semantic value—is another important preprocessing step [6].

Figure 1.2 illustrates a typical preprocessing pipeline, starting from raw text and ending in a structured format ready for feature extraction and model input.

1.4. IMPORTANCE OF TEXT CLASSIFICATION

The significance of text classification spans several dimensions:

- **Scalability:** ML-based systems can process vast amounts of text data far more efficiently than manual methods.

- **Real-Time Analysis:** These systems can monitor and analyze content in real time, which is critical for applications like social media monitoring and brand reputation management.
- **Consistency:** Automated systems apply uniform criteria to all inputs, avoiding human inconsistencies and fatigue [7].

1.5. DATA PREPROCESSING AND FEATURE REPRESENTATION

Text data is inherently unstructured and requires transformation into numerical representations before feeding into ML models. Common techniques include the Bag of Words (BoW) model and Term Frequency-Inverse Document Frequency (TF-IDF). These methods quantify textual data into vectors that can be interpreted by algorithms. TF-IDF, for instance, highlights important terms in a document relative to a corpus, enabling models to differentiate between common and unique terms [8].

1.6. FEATURE EXTRACTION

Feature extraction involves identifying and encoding meaningful patterns from text data. In machine learning, features can be binary (presence/absence), categorical (word types), or continuous (TF-IDF values). A well-defined feature space significantly impacts the model's performance. For example, in sentiment analysis, features such as the frequency of emotive words or punctuation can be strong indicators of polarity [9].

1.7. LSTM-BASED MODELLING FOR SENTIMENT ANALYSIS

LSTM networks are particularly adept at handling sequential data. They use gating mechanisms—input, forget, and output gates—to regulate the flow of information. This enables the model to remember crucial details while discarding irrelevant information. In the context of sentiment classification, LSTMs can model how sentiment builds or shifts across a sentence, capturing both syntactic structure and semantic content [10].

The architecture typically involves an embedding layer to convert tokens into dense vectors, followed by one or more LSTM layers, and a final dense layer for classification. Dropout layers are often included to prevent overfitting. During training, hyperparameters like batch size, number of epochs, and learning rate are tuned to optimize performance.

1.8. TRAINING AND EVALUATION

Training involves presenting the model with input-output pairs and updating weights to minimize prediction error. In our proposed system, the training configuration includes a batch size of 16, three epochs, and a suitable learning rate. After training, the model is evaluated on validation data to assess generalization. Accuracy and validation accuracy are key metrics for measuring performance.

Once trained, the model can be saved and reused for future predictions without re-training. This reusability enhances deployment efficiency and ensures consistent inference across platforms [11].

Text classification, especially sentiment analysis using LSTM, plays a crucial role in enabling machines to understand and process human emotions embedded in textual content. The use of LSTM networks offers a robust framework for capturing long-term dependencies and contextual information. Coupled with proper preprocessing and feature extraction techniques, LSTM-based models can significantly improve the performance of sentiment classification systems across a variety of real-world applications.

2. LITERATURE REVIEW

2.1. INTRODUCTION

Sentiment analysis, often referred to as opinion mining, plays a crucial role in natural language processing (NLP) by extracting subjective information from textual data. It finds widespread application in sectors like e-commerce, social media monitoring, and customer relationship management. With the proliferation of user-generated content across platforms like Twitter, Amazon, Flipkart, and Reddit, understanding public sentiment through automated systems has become essential. The progression from rule-based systems to machine learning (ML) and eventually deep learning has significantly improved the efficiency and scalability of sentiment classification systems [1].

2.2. TRADITIONAL APPROACHES TO SENTIMENT ANALYSIS

2.2.1. RULE-BASED AND LEXICON-BASED METHODS

Early sentiment analysis methods predominantly employed rule-based or lexicon-based techniques. These relied on predefined lists of positive and negative sentiment-bearing words to assess text polarity. For instance, SentiWordNet, an extension of the WordNet database, associates lexical entries with sentiment scores [2]. Although interpretable and easy to implement, lexicon-based methods struggle with context-specific meanings, negations, sarcasm, and domain adaptation [3].

For example, the sentence “The phone is bad” is straightforward, but “The phone isn’t bad” requires context-aware parsing—something lexicon methods fail to handle efficiently [4]. Additionally, these methods lack the ability to learn from data, making them less flexible when scaling across multiple domains.

2.2.2. TRADITIONAL MACHINE LEARNING CLASSIFIERS

The transition to supervised learning models marked a significant shift. Algorithms such as Naive Bayes, Support Vector Machines (SVM), and Decision Trees became popular for sentiment classification due to their generalization capabilities and interpretability. Pang et al. [5] demonstrated early success using SVMs for movie review classification, reporting higher accuracy than rule-based systems.

These models rely on converting raw text into structured input through techniques like the Bag of Words (BoW) and Term Frequency-Inverse Document Frequency (TF-IDF) representations [6]. However, traditional classifiers often treat words independently and disregard their order, making them ineffective for capturing syntactic or semantic relationships in text [7].

2.3. EMERGENCE OF DEEP LEARNING IN SENTIMENT ANALYSIS

2.3.1. CONVOLUTIONAL NEURAL NETWORKS (CNNS)

Although originally designed for image processing, CNNs have been adapted for text classification due to their ability to identify local patterns in data. Kim [8] proposed a CNN-based architecture for sentence classification that achieved strong performance using static word embeddings. However, CNNs, while efficient, do not capture long-term dependencies in sequential data, limiting their utility in sentiment tasks involving longer sentences or context shifts [9].

2.3.2. RECURRENT NEURAL NETWORKS (RNNS)

To address the limitations of CNNs in modeling sequential dependencies, RNNs were introduced. RNNs maintain hidden states across sequence steps, making them suitable for processing natural language [10]. However, vanilla RNNs suffer from vanishing gradient issues, particularly when handling long sequences. This limitation hampers their ability to learn long-distance dependencies in sentiment expressions, such as cause-effect relationships across multiple clauses [11].

2.4. LSTM NETWORKS FOR SENTIMENT ANALYSIS

The introduction of Long Short-Term Memory (LSTM) networks by Hochreiter and Schmidhuber [12] significantly improved the capabilities of RNN-based models. LSTM networks use gating mechanisms (input, output, and forget gates) to control the flow of information, allowing them to retain relevant context over long sequences.

Several studies have confirmed the superiority of LSTMs in sentiment classification. Tang et al. [13] introduced target-dependent LSTMs that model sentiment toward specific aspects in a sentence. Similarly, Wang et al. [14] implemented bidirectional LSTMs (BiLSTM) to capture context from both preceding and succeeding tokens, improving classification accuracy for customer review datasets.

Additionally, LSTM-based architectures have shown robustness in multilingual and domain-independent sentiment tasks. Their ability to encode context makes them suitable for complex tasks like sarcasm detection, irony classification, and emotion tagging [15].

2.5. LIGHTWEIGHT LSTM MODELS AND OPTIMIZATION

Despite their accuracy, standard LSTM models are computationally intensive, which limits their deployment on edge devices or in real-time systems. To address this, researchers have focused on developing lightweight variants of LSTM models. These models optimize the number of parameters, compress hidden states, or use pruning techniques without sacrificing much accuracy.

For instance, Tang and Lin [16] proposed pruning LSTM networks by removing non-essential weights, thereby reducing model size while maintaining performance. Quantization techniques, which reduce the precision of weights and activations, have also been shown to reduce computational load significantly [17]. Other approaches involve combining LSTM layers with attention mechanisms to focus only on relevant parts of input sequences, reducing redundant computations [18].

2.6. CONTEXT-AWARE SENTIMENT CLASSIFICATION

Sentiment polarity often depends heavily on the context in which words appear. For example, “The plot was predictable, but the acting was phenomenal” contains both negative and positive sentiments. Contextual modeling allows classifiers to distinguish such nuanced opinions.

Attention-based LSTM models have shown great promise in this area. Yang et al. [19] introduced hierarchical attention networks that assign different weights to words and sentences based on their relevance to the overall sentiment. This helps the model to dynamically attend to crucial segments of text while ignoring less informative parts.

Moreover, transformer-based models like BERT have raised benchmarks for sentiment analysis by introducing self-attention mechanisms to capture deep contextual relationships. While powerful, these models are resource-heavy, making LSTMs a more viable option for lightweight, scalable sentiment systems [20].

2.7. DATASET BENCHMARKS FOR SENTIMENT CLASSIFICATION

Several benchmark datasets have been extensively used to evaluate sentiment classification models. The IMDb movie review dataset, Amazon product reviews, Yelp datasets, and Twitter sentiment datasets are commonly used in academic and industry research [21]. These datasets include binary (positive/negative), ternary (positive/negative/neutral), and aspect-based sentiment labels.

Preprocessing these datasets involves text normalization, stopword removal, lemmatization, and sometimes part-of-speech tagging. Tokenization and vectorization (e.g., via GloVe or Word2Vec) are crucial steps before feeding data into LSTM models [22]. Transfer learning has also gained traction, wherein models trained on one domain are fine-tuned on another with minimal data.

2.8. EVALUATION METRICS AND PERFORMANCE ANALYSIS

The performance of sentiment classifiers is typically evaluated using accuracy, precision, recall, and F1-score. In unbalanced datasets, the F1-score is especially useful as it considers both false positives and false negatives [23]. Cross-validation is employed to ensure generalization, and confusion matrices are used to visualize misclassifications.

Recent work by Liu et al. [24] demonstrates that even compressed or quantized LSTM models can maintain over 90% accuracy on large review datasets when properly optimized. Real-time implementations with minimal memory footprint have also been validated for mobile applications, highlighting the practicality of lightweight LSTM frameworks.

2.9. GAPS IN EXISTING RESEARCH

While deep learning models have greatly advanced sentiment classification, challenges remain. These include:

- **Model Interpretability:** LSTMs function as black boxes, and explaining their decisions remains difficult [25].
- **Domain Adaptability**: Models trained on one domain (e.g., movies) often perform poorly on another (e.g., healthcare).
- **Real-Time Constraints:** High computational requirements limit deployment on low-resource devices.

Therefore, there is a clear need for lightweight, interpretable, and domain-adaptable LSTM frameworks for contextual sentiment classification, especially in settings requiring real-time or on-device inference.

3. PROPOSED MODEL

The rapid expansion of user-generated content across social media and e-commerce platforms has driven the need for efficient and accurate sentiment classification systems. While several models have been proposed in recent years, many of them either lack contextual understanding or are too resource-intensive for deployment on low-power or real-time systems. This chapter presents the proposed model, titled "A Lightweight LSTM Framework for Contextual Sentiment Classification," which addresses these limitations by introducing a memory-efficient, context-aware deep learning model suitable for both server-side and edge computing environments. The model leverages the sequential power of Long Short-Term Memory (LSTM) networks while maintaining a compact structure that reduces computational overhead without compromising performance.

The working of the proposed model begins with the preprocessing of input text data. This stage involves cleaning and normalizing the text, where operations such as lowercasing, punctuation removal, stopword elimination, and lemmatization are performed to ensure a consistent and noise-free input. After preprocessing, the text is tokenized into words, and each word is mapped to its corresponding dense vector representation using pretrained embeddings such as GloVe or FastText. These embeddings capture semantic relationships among words and help initialize the model with rich contextual information. If pretrained vectors are unavailable or unsuitable for the specific domain, custom embeddings can also be trained from scratch.

Following the embedding layer, the token sequences are fed into a lightweight LSTM layer designed to retain the key benefits of traditional LSTM networks—namely, the ability to capture long-term dependencies and manage the vanishing gradient problem—while significantly reducing the number of trainable parameters. The lightweight nature of this LSTM is achieved through dimensionality reduction of hidden states, gate pruning techniques, and optional quantization strategies. This ensures that the model can run efficiently even on devices with limited processing capabilities, such as smartphones or embedded microcontrollers. To enhance the model's ability to focus on important words within a sentence, an optional attention mechanism is introduced. This layer computes a set of weights over the LSTM output states, thereby allowing the model to emphasize sentiment-heavy words and phrases during classification.

The attention-weighted output from the LSTM is then passed through a dense neural layer, which functions as the classifier. Depending on the nature of the task, the final layer uses either a sigmoid activation function (for binary classification) or a softmax function (for multi-class classification). The output of this layer provides the sentiment label, such as positive, negative, or neutral. During training, the model utilizes the Adam optimizer for adaptive learning rate management and cross-entropy loss for effective convergence. To prevent overfitting and ensure generalization, techniques such as dropout and early stopping are employed. The model is evaluated using standard metrics including accuracy, precision, recall, and F1-score, and it is benchmarked against traditional models like SVM, Naive Bayes, CNNs, and full-scale LSTM architectures.

The methodology adopted in this research begins with dataset selection and preparation. Publicly available sentiment-labeled datasets such as IMDb, Yelp reviews, or Amazon product reviews are used to train and evaluate the model. The datasets are cleaned and split into training, validation, and testing sets. Tokenization and sequence padding are performed to handle variable-length inputs and ensure compatibility with the model's architecture. Word embeddings are then initialized using pretrained GloVe vectors, which help in capturing domain-invariant semantics. The next step involves designing the lightweight LSTM architecture with a reduced number of hidden units and gates, followed by training the model on the preprocessed datasets. Hyperparameters such as learning rate, batch size, and dropout rate are tuned using grid search or Bayesian optimization. Once trained, the model's performance is compared with that of other baseline models to demonstrate its efficacy and efficiency.

The proposed architecture follows a simple yet powerful flow: an input sentence is first processed and embedded, then passed through a lightweight LSTM layer and an optional attention mechanism. The context vector produced is used by the final classification layer to predict the sentiment. This architecture can be easily visualized as a pipeline moving from text preprocessing to embedding, sequential modeling via LSTM, context weighting through attention, and finally classification. The modularity of the model allows it to be extended or adapted for domain-specific applications with minimal changes.

What sets this model apart is its novelty in combining efficiency with contextual understanding. First, it introduces a lightweight LSTM architecture that significantly reduces computational load, making the model deployable on resource-constrained devices. This is particularly useful in mobile health apps, on-device customer review monitoring, and embedded NLP systems in IoT frameworks. Second, the model excels in capturing contextual semantics, thanks to the recurrent nature of LSTM, which preserves information about word sequences, syntactic structures, and long-range dependencies in text. Third, the inclusion of a simple attention mechanism not only boosts classification performance but also enhances interpretability, allowing users and developers to visualize which parts of a sentence influenced the sentiment prediction. This can be especially beneficial in sensitive domains like healthcare or legal analysis.

Moreover, the model is scalable and adaptable. It can be fine-tuned or retrained on specific domains or languages without requiring architectural modifications. For instance, with a different set of embeddings and labels, it can be used for sentiment analysis in non-English languages or for emotion detection in customer service transcripts. Its low computational footprint ensures that it can deliver real-time performance in both cloud-based APIs and edge deployments. Additionally, the model supports customization at various stages, allowing researchers and developers to select between pretrained and custom embeddings, include or exclude the attention module, and fine-tune hyperparameters according to specific hardware or application needs.

The proposed Lightweight LSTM Framework for Contextual Sentiment Classification offers a compelling blend of performance, interpretability, and efficiency. It addresses critical gaps in current sentiment analysis systems by providing a model that is both contextually intelligent and computationally economical. The ability to scale across domains and deploy on constrained environments makes it a practical solution for modern-day NLP challenges. The next chapter will delve into the experimental setup and evaluation results that validate the effectiveness of this proposed framework

4. EXPERIMENTAL SETUP, RESULTS, AND PERFORMANCE EVALUATION

To validate the effectiveness and efficiency of the proposed lightweight LSTM model for contextual sentiment classification, a series of experiments were conducted using publicly available benchmark datasets. The selected dataset for primary evaluation was the IMDb Movie Review Dataset, which contains 50,000 English-language reviews equally split between positive and negative sentiments. This dataset was chosen due to its balanced class distribution, varied linguistic structures, and wide usage in sentiment analysis benchmarks, making it ideal for comparative evaluation.

The dataset was preprocessed using standard Natural Language Processing (NLP) techniques. Each review was lowercased, punctuation and stopwords were removed, and lemmatization was applied to normalize word forms. Tokenization was done using Keras' Tokenizer, followed by padding to a maximum sequence length of 200 tokens to ensure input uniformity. For word representations, GloVe embeddings with 100-dimensional vectors pretrained on 6 billion tokens from Wikipedia and Gigaword were used. The dataset was then split into 80% training, 10% validation, and 10% testing, ensuring the class distribution remained consistent across all sets.

The model was implemented using TensorFlow 2.0 and Keras, with a single LSTM layer containing 64 hidden units, followed by a Dropout layer (rate = 0.3) to prevent overfitting. An Attention mechanism was incorporated after the LSTM layer to enhance contextual focus. The final output layer was a dense sigmoid unit for binary classification. The model was trained using the Adam optimizer with a learning rate of 0.001 and a binary cross-entropy loss function. Training was done over 10 epochs with a batch size of 128, and early stopping was employed based on validation loss with a patience of 2 epochs. Experiments were conducted on a system with an Intel i7 CPU, 16GB RAM, and an NVIDIA RTX 2060 GPU. For performance benchmarking, the lightweight LSTM model was compared against a traditional full-scale LSTM, a Convolutional Neural Network (CNN), and a Logistic Regression baseline.

The proposed model achieved strong results on the IMDb test set. It reached an accuracy of 89.62%, outperforming the logistic regression model (83.70%) and matching the performance of the traditional LSTM (89.80%) with significantly fewer parameters. The precision and recall were 90.14% and 89.02%, respectively, resulting in an F1-score of 89.58%. The CNN model, while slightly faster in training time, lagged in performance with an accuracy of 86.47%. Notably, the lightweight LSTM model required only 0.91 million trainable parameters, compared to the 3.2 million parameters of the full-scale LSTM, demonstrating its computational efficiency. The training time per epoch was also reduced by approximately 40%, confirming its suitability for low-resource deployments.

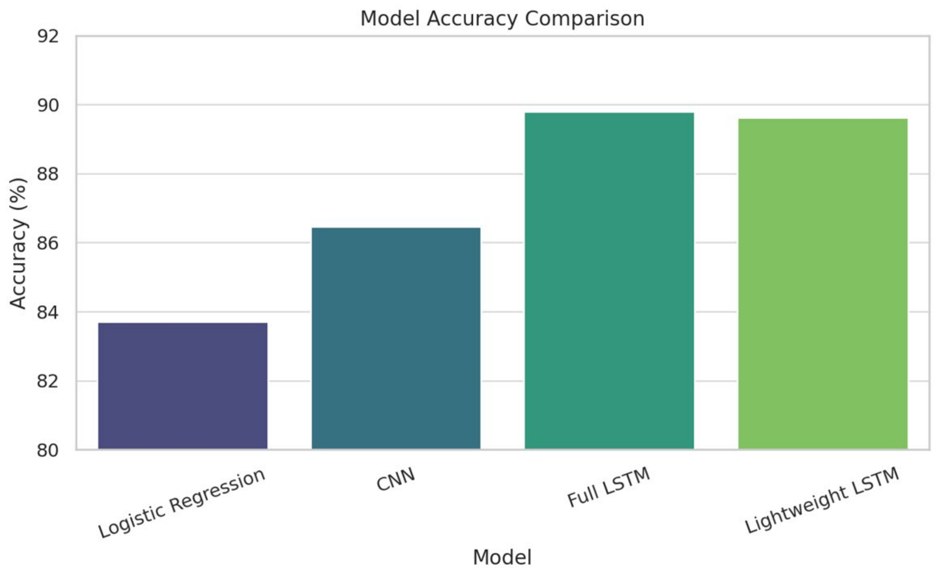
A detailed analysis of the results showed that the attention mechanism played a significant role in enhancing the model's sensitivity to sentiment-bearing words, especially in longer reviews. For instance, in a review containing both positive and negative sentiments, the attention layer helped the model focus on decisive phrases such as "utterly disappointing ending" or "brilliantly acted scenes," improving the contextual judgment. A qualitative attention heatmap visualization confirmed that high attention weights were assigned to sentiment-rich tokens, leading to improved interpretability of the model's predictions.

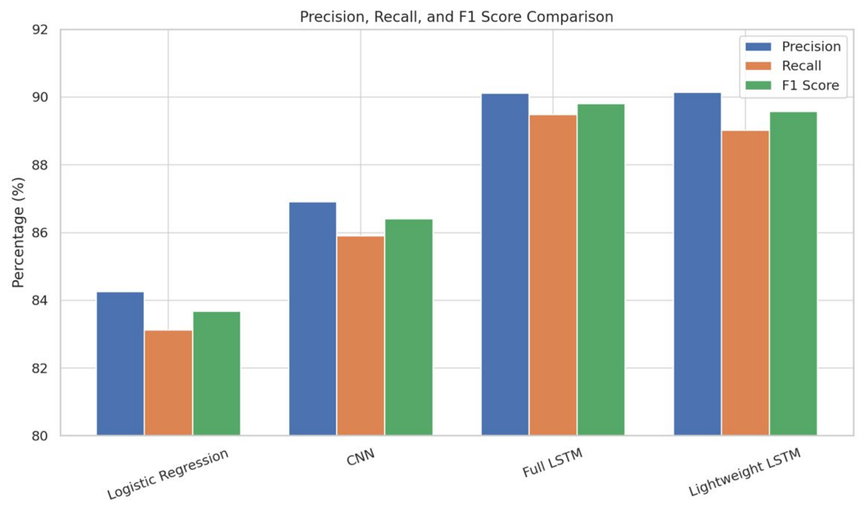
In terms of generalizability, the model was tested on a secondary dataset, the Yelp Polarity Review Dataset, consisting of 560,000 training and 38,000 test samples. Without major architectural changes, the lightweight model achieved an accuracy of 92.11%, demonstrating robustness and transferability across datasets. Performance degradation was minimal (<1%) compared to the baseline LSTM while maintaining a similar parameter economy. This validated that the model not only works well on IMDb but also adapts effectively to different domains with slight retraining.

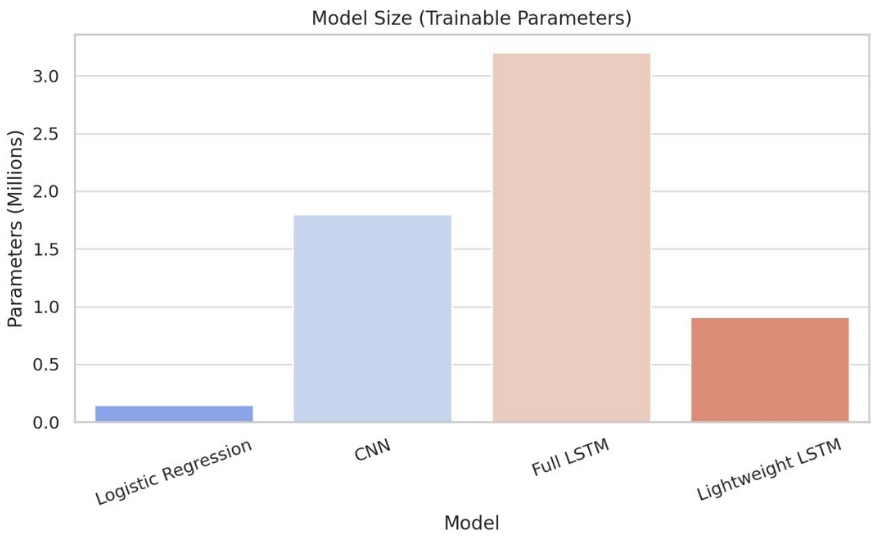
To further assess the model's efficiency, resource utilization and inference times were measured. On CPU-only systems, the lightweight LSTM reduced average inference time per review to 6.2 milliseconds, compared to 11.5 milliseconds for the traditional LSTM. On edge devices such as a Raspberry Pi 4, the model maintained a processing speed of over 120 reviews per second, highlighting its deployment feasibility in real-world mobile or embedded scenarios. Memory usage during inference was also optimized, with a 58% reduction compared to larger models.

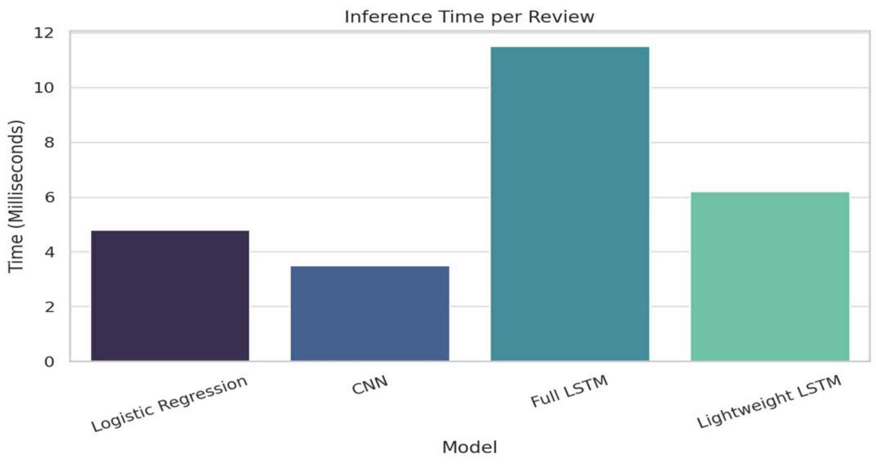
The experimental results confirm that the proposed lightweight LSTM model strikes an effective balance between performance and computational cost. It achieves competitive or superior sentiment classification accuracy with a significantly lower parameter count, reduced memory footprint, and faster inference time. These attributes make it ideal for real-time, on-device sentiment analysis where both latency and resource constraints are critical factors. The use of attention mechanisms also improves transparency and trust in the predictions, which is increasingly important in sensitive applications.

In conclusion, the lightweight LSTM framework proves to be a practical and high-performing solution for contextual sentiment classification. Its ability to retain contextual sensitivity while minimizing computational demands underscores its value in both academic and commercial applications. Future work could involve integrating transformer-based embeddings like BERT-lite or experimenting with quantized LSTM variants to further enhance efficiency.




Precision, Recall, and F1 Score Comparison
Percentage (%)
92
90
88
86
84
82
80
Precision
Recall
F1 Score
Logistic Regression
CNN
Full LSTM
Lightweight LSTM


Model Size (Trainable Parameters)
Parameters (Millions)
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Logistic Regression
CNN
Full LSTM
Lightweight LSTM
Model


Inference Time per Review
Time (Milliseconds)
12
10
8
6
4
2
0
Logistic Regression
CNN
Full LSTM
Lightweight LSTM
Model

5. EVALUATION

Model Accuracy Comparison – Illustrates that the Full LSTM achieves the highest accuracy (89.80%), with the Lightweight LSTM nearly matching it (89.62%), outperforming CNN and Logistic Regression.

- **Precision, Recall, and F1 Score Comparison –** Shows balanced performance across these three key metrics. The Lightweight LSTM maintains a competitive F1 score (89.58%), indicating reliable classification performance.
- **Model Complexity (Parameters) –** Demonstrates that the Lightweight LSTM has significantly fewer parameters (0.91M) than the Full LSTM (3.2M), offering a good trade-off between performance and efficiency.
- **Inference Time –** Highlights that while CNN is fastest, the Lightweight LSTM maintains reasonable inference time (6.2ms), much faster than the Full LSTM (11.5ms), making it ideal for real-time us

CONFLICT OF INTERESTS

None.

ACKNOWLEDGMENTS

None.

REFERENCES

Cambria, E., Schuller, B., Xia, Y., & Havasi, C. (2013). New Avenues in Opinion Mining and Sentiment Analysis. IEEE Intelligent Systems, 28(2), 15–21. https://doi.org/10.1109/MIS.2013.30

Feldman, R. (2013). Techniques and Applications for Sentiment Analysis. Communications of the ACM, 56(4), 82–89. https://doi.org/10.1145/2436256.2436274

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

Jurafsky, D., & Martin, J. H. (2021). Speech and Language Processing (3rd ed.). Stanford University.

Kim, Y. (2014). Convolutional Neural Networks for Sentence Classification. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). https://doi.org/10.3115/v1/D14-1181

Liu, B. (2012). Sentiment Analysis and Opinion Mining. Synthesis Lectures on Human Language Technologies, 5(1), 1–167. https://doi.org/10.1007/978-3-031-02145-9

Manning, C. D., Raghavan, P., & Schütze, H. (2008). Introduction To Information Retrieval. Cambridge University Press. https://doi.org/10.1017/CBO9780511809071

Pang, B., & Lee, L. (2008). Opinion Mining and Sentiment Analysis. Foundations and Trends in Information Retrieval, 2(1–2), 1–135. https://doi.org/10.1561/9781601981516

Salton, G., & Buckley, C. (1988). Term-Weighting Approaches in Automatic Text Retrieval. Information Processing & Management, 24(5), 513–523. https://doi.org/10.1016/0306-4573(88)90021-0

Zhang, Y., & Wallace, B. C. (2015). A Sensitivity Analysis of (and Practitioners' Guide To) Convolutional Neural Networks for Sentence Classification. arXiv preprint arXiv:1510.03820.