Mining Text for Insight - Yelp Review Analysis

Akanksha Shah
Texas A&M University
College Station, Texas, USA
shahakanksha@tamu.edu

Arvinder Singh Mundra
Texas A&M University
College Station, Texas, USA
arvinder.mundra@tamu.edu

Kyren Liu
Texas A&M University
College Station, Texas, USA
qil012@tamu.edu

Tasfin Mahmud
Texas A&M University
College Station, Texas, USA
tasfinmahmud@tamu.edu

***Abstract*—Understanding how written opinions translate into numerical ratings is an important challenge in sentiment and review analysis. Platforms that host large volumes of user generated feedback, such as Yelp, largely benefit from models that can infer ratings directly from text, eliminating their dependency on user-provided stars and enabling deeper behavioral insight. In this work, we conduct a detailed examination of the Yelp Review Full dataset and build a complete text-processing pipeline involving cleaning, normalization, tokenization, vocabulary construction, and sequence padding. We then design and evaluate two neural architectures: a bidirectional LSTM model and a hybrid CNN–BiLSTM model that integrates convolutional feature extraction with recurrent temporal modeling. Both models are trained and assessed under identical experimental settings using accuracy, precision, recall, and F1-score as evaluation metrics. Our findings indicate that while both models perform reliably, the added complexity of the CNN–BiLSTM does not yield substantial improvement over the simpler BiLSTM baseline. These experiments highlight the strength of LSTM-based methods for rating prediction and suggest that architectural complexity alone does not guarantee performance gains for this task. [Project Website]**

***Index Terms*—Multiclass Text Classification, Rating Prediction, Yelp Reviews, Sentiment Analysis, Bidirectional LSTMs, Hybrid CNN–BiLSTM, Natural Language Processing.**

I. Introduction

Nowadays, online review platforms such as Yelp have become a primary choice for people to decide where to eat, travel, or shop. In this project, we study document-level sentiment classification on the Yelp Full Review dataset, which contains 700k reviews labeled with ratings from 0 as worst to 4 as best. We follow the standard split of 650k training with validation reviews and 50k test reviews. Since the rating distribution is balanced across classes, with each class representing 20 percent of the data, we focus more on model behavior and comparative performance rather than on avoiding bias introduced by class imbalance.

After data wrangling and EDA, we examined the structure of the feature space, then apply TF-IDF representations and applied singular value decomposition (SVD) to project the reviews in a lower-dimensional space. The reduced representations showcased a substantial overlap between the rating classes, suggesting that the classes are not easily separable based on surface-level lexical patterns. These initial observations motivated to conduct experiments with deep sequence architectures as BiLSTM and a hybrid CNN-BiLSTM model.

We started our project by placing our data into prior research on document-level sentiment analysis and neural text models, and among all relevant articles, Rao and Belaroussi's product inspired us the most. Rao's research strengthened our decision to use LSTM models because Rao points out that one of the main challenges in document-level sentiment analysis is managing long reviews that get different opinions. In his article, he states on single-layer LSTMs can't mention both on sentence details and on big picture across the analysis. Their resolution suggests SR-LSTM as a two-layer model for the first LSTM converts every sentence into fixed-length vector, while the second LSTM makes them as a whole. Nevertheless, SSR-LSTM, another model that uses a sentiment dictionary to filter out non-emotional sentences before sending them through the network. On datasets like IMDB and Yelp in 2014 and 2015, these models behave better than methods like SVM and Naive Bayes, and even earlier RNNs [1]. These model specialties point out the value of modeling sentence structure and the connections between reviews, which enlightened our idea of LSTM-based architectures for sentiment analysis on our Yelp reviews dataset.

Meanwhile, Belaroussi used LSTM-based models and compare it with transformer models like BERT, RoBERTa, BERTweet, and DistilBERT. These models all perform on a three-way Yelp sentiment task as negative, neutral, and positive. Belaroussi pre-trained Hugging Face models and find that transformers focus more on their accuracy and applications. But ironically the high complexity BERT models don't enhance better performance than those beginner models, so using bidirectional LSTMs doesn't guarantee success. Their results show that with the right tuning, LSTMs can still hold their own against transformers and sometimes even overcome strong baselines [2]. From this perspective, we design our own LSTM models by getting the balance between how complex the model is and how well it works.

II. Methodology

This section describes the experimental setup and methodology, including the data preprocessing steps applied to get meaningful information from the Yelp reviews. Additionally, we describe the implementation model, detailing its architecture, feature embeddings, and the training and validation processes. To investigate the impact of different sequence-modeling strategies, we implement two deep learning architectures. The first is a Bidirectional Long Short-Term Memory (BiLSTM) network, which captures contextual dependencies in both forward and backward directions of the review text.

The second is a hybrid CNN–LSTM model that combines convolutional layers for local n-gram feature extraction with an LSTM layer to model longer-range textual patterns. These two architectures allow us to compare traditional recurrent sequence modeling with a convolution-augmented approach under identical experimental conditions. Finally, we discuss the evaluation strategy that is used to measure the model performance, ensuring a controlled comparison between the two neural architectures.

A. *Data Preprocessing and Tokenization*

The raw reviews in the dataset contain noise and inconsistencies, so the text is cleaned before modeling. All rows with null or empty reviews are removed, and the remaining text is normalized by lowercasing and replacing escape characters such as “\n”, “\r”, and stray backslashes with spaces. URLs are replaced with a `<URL>` token to reduce variability across reviews. Contractions (e.g., “don’t”) are preserved to maintain accurate expression of negation. Unlike traditional TF–IDF or bag-of-words models where stop words are typically removed, we retain them because LSTM-based architectures rely on the full sentence structure to learn contextual and syntactic patterns; removing function words, especially negation terms such as “not,” can significantly alter the implied meaning. Emoticons are converted into `<POS_EMOTICON>` and `<NEG_EMOTICON>` tokens, to consistently capture emotional cues. After normalization, the reviews are tokenized using a whitespace-based tokenizer that preserves words and special tokens. For example, “The service was not good :(!!” becomes `["the, "service", "was", "not", "good", "<NEG_EMOTICON>", "!"]`. These token sequences are then used for vocabulary construction & encoding.

B. *Vocabulary Construction*

Following preprocessing and tokenization, a word-level vocabulary is built solely from the training set, assigning a distinct integer index to every token. Let $\mathcal{T} = \{t_1, t_2, \ldots, t_N\}$ denote the set of all unique tokens identified in the processed training data, and let $f(t_i)$ represent the frequency of token t_i. The tokens that appear less than five times (i.e., $f(t_i) < 5$) are omitted to minimize noise, reduce input dimensionality, and keep model complexity manageable. The remaining tokens are sorted in descending order of frequency, and the vocabulary is capped at $|\mathcal{V}| = 20{,}000$ words, which is an empirical limit determined to balance accuracy and computational efficiency.

The vocabulary is then defined as a mapping $v : \mathcal{T} \rightarrow \{0, 1, \ldots, |\mathcal{V}| - 1\}$, with reserved indices for special tokens such as `<PAD>` and `<UNK>` which are used to handle padding and out-of-vocabulary words. This vocabulary serves as backbone for all subsequent encoding and modeling components.

C. *Sequence Encoding and Padding*

Once the vocabulary is built, each tokenized review is transformed into a sequence of integer indices based on the vocabulary mapping. Let $[t_1, t_2, \ldots, t_n]$ represent the sequence of tokens. Each token is replaced with its corresponding index from the vocabulary, resulting in the encoded sequence $[v(t_1), v(t_2), \ldots, v(t_n)]$, where $v(\cdot)$ indicates the index lookup in the vocabulary. Tokens that are not present in the vocabulary are substituted with the index of a reserved special token, `<UNK>`. To support efficient batching and ensure consistent input dimensions to the model, all encoded sequences are standardized to a fixed length L. Based on experimental results and EDA of review length, 99% of reviews fall below 684 words; therefore, we chose $L = 650$. Review lengths shorter than L are padded with the `<PAD>` special token, while those longer than L are truncated. This input representation is then used by both the BiLSTM and CNN–LSTM model.

D. *BiLSTM Model Architecture*

Long Short-Term Memory (LSTM) networks are a recurrent neural network variant designed to mitigate vanishing and exploding gradients through gated units including input, forget, and output gates. These determine how information travels through the network over time. These gates enable the model to determine when to write new information, when to remember or forget past information, and when to expose internal memory to the next layer. This architecture enables them to model long-term dependencies and keep important contextual information throughout the long review sequences.

A Bidirectional LSTM extends this capability by running two independent LSTMs in both forward and backward directions. The forward LSTM captures context from earlier tokens, whereas the backward LSTM captures information from later ones. Concatenating the hidden states from both directions results in a more detailed and complete representation of each review. This bidirectional processing is useful for rating prediction because key sentiment cues (the user’s opinion) may occur anywhere in the review text [3].

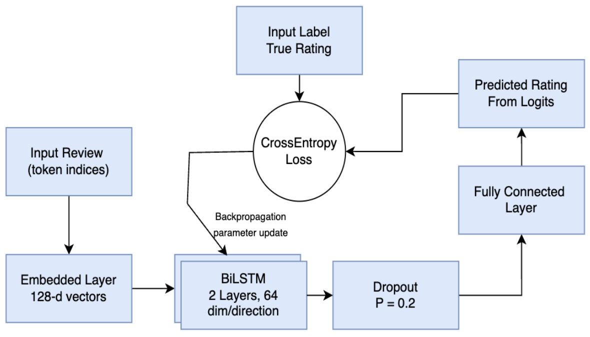


Fig. 1: BiLSTM Model Architecture

The model begins with an embedding layer that turns each token index into a 128-dimensional vector, allowing the network to learn distributed representations of words. These embeddings are used as input for a stacked, two-layer bidirectional LSTM with 64 hidden units in each direction. Stacking layers allows the model to create increasingly abstract features, and bidirectionality ensures that contextual information from the entire review influences the representation.

A dropout rate of 0.2 is applied between LSTM layers to reduce overfitting and improve generalization. Following the last recurrent layer, the model recovers the final forward and backward hidden states, which are concatenated into a single vector that summarizes the entire review. This representation captures both the overall sentiment flow and the detailed language structure throughout the review.

Finally, this summary vector is sent into a fully connected linear layer, which maps it to the five Yelp rating classes. The output logits indicate the model's confidence at each rating level, and the projected class is determined by selecting the index with the highest value. We train the model using cross-entropy loss with learning rate 2×10^{-3}, weight decay 10^{-5}, and batch size 64.

E. CNN–BiLSTM Model Architecture

We also implement a CNN-BiLSTM model to compare with the BiLSTM one for this text classification problem, as proposed in the articles for classifying Reuters-21578 dataset and pilgrim reviews [4], [5]. Each input sequence has a fixed length of 650 tokens, and every token is mapped to a 64-dimensional embedding vector. This produces an embedding matrix $E \in \mathbb{R}^{650\times 64}$.

To capture short and meaningful local patterns, we apply a 1-D convolution layer with 64 filters of kernel size 3. For a filter w and bias b, the convolution output at position t is

$$h_t = \mathrm{ReLU}(w * E_{t:t+2} + b). \tag{1}$$

A max-pooling operation with stride 2 reduces the sequence length by half. The pooled features are then passed into a two-layer bidirectional LSTM with hidden size 64 per direction. For each time step t, the LSTM computes the following:

$$\begin{aligned} f_t &= \sigma\big(W_f[h_{t-1}, x_t] + b_f\big), & i_t &= \sigma\big(W_i[h_{t-1}, x_t] + b_i\big), \\ o_t &= \sigma\big(W_o[h_{t-1}, x_t] + b_o\big), & \tilde{c}_t &= \tanh\big(W_c[h_{t-1}, x_t] + b_c\big), \\ c_t &= f_t \odot c_{t-1} + i_t \odot \tilde{c}_t, & h_t &= o_t \odot \tanh(c_t). \end{aligned} \tag{2}$$

Here, x_t is the input at time t, h_{t-1} is the previous hidden state, f_t is the forget gate, i_t is the input gate, o_t is the output gate, $\tilde{c}_t$ is the candidate cell update, c_t is the memory cell, and h_t is the updated hidden state. The sigmoid function $\sigma(\cdot)$ controls how information flows, while $\odot$ denotes elementwise multiplication.

We concatenate the final forward and backward hidden states to obtain a 256-dimensional representation. A dropout layer with probability 0.3 is applied to reduce overfitting. Finally, a fully connected layer maps the representation to a 5-dimensional output vector: $y = Wh + b$.

We train the model using cross-entropy loss with learning rate 5×10^{-4}, weight decay 10^{-5}, and batch size 32.

F. Training, Validation and Evaluation

We trained our models using a total of 700,000 labeled Yelp reviews. The dataset was divided into 520,000 samples for training, 130,000 for validation, and 50,000 for testing, corresponding to an 80/20 split of the original training set followed by a held-out test set. All splits preserved the original class distribution through stratified sampling. During training, we processed mini-batches and optimized the parameters using the Adam optimizer, which adapts the learning rate by maintaining first and second moment estimates of the gradients. Considering y_c the true label and $\hat{y}_c$ the predicted probability for class c, the objective function for model training was the multi-class cross-entropy loss, defined as

$$\mathcal{L}_{CE} = -\sum_{c=1}^{C} y_c \log(\hat{y}_c), \tag{3}$$

After each epoch, we evaluated the performance of the model on the validation set. The weights were restored according to the model with the minimum validation loss.

We evaluated the model on the 50,000-sample unseen test set at the final stage to compute the performance metrics, as follows:

$$\begin{aligned} \text{Acc} &= \frac{TP + TN}{TP + TN + FP + FN}, & \text{Prec} &= \frac{TP}{TP + FP}, \\ \text{Rec} &= \frac{TP}{TP + FN}, & \text{F1} &= 2\,\frac{\text{Prec} \cdot \text{Rec}}{\text{Prec} + \text{Rec}}. \end{aligned} \tag{4}$$

Where TP, FP, TN, and FN represent True Positives, False Positives, True Negatives and False Negatives, respectively. These metrics provide a detailed The above-mentioned metrics provide us with insight regarding performance differences across classes.

III. Experimental Results and Discussion

This section presents the experimental findings of our baseline BiLSTM model, and the additional experiments conducted using the CNN-BiLSTM architecture for comparative analysis.

A. Effect of Vocabulary Size and Maximum Sequence Length

To investigate how input representation affects model performance, we trained and optimized the BiLSTM model with different vocabulary sizes (20k, 30k, 40k) and maximum sequence lengths (128, 256, 512, and 650). To ensure a fair comparison, hyperparameters for each configuration were tuned separately, and the highest test accuracy was noted.

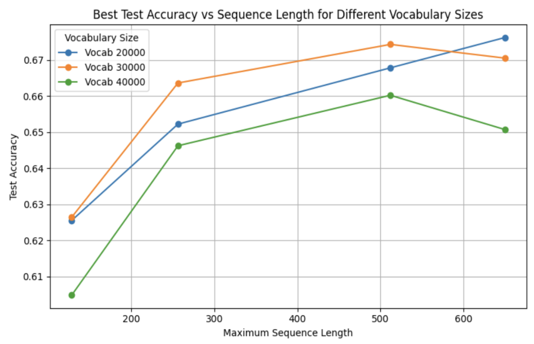


Fig. 2: Variation of Test Accuracy with Different Vocab Sizes and Max Sequence Length

Figure 2 shows a clear pattern across all configurations. Increasing the maximum sequence length results in a significant gain in accuracy. Accuracy increases dramatically while advancing from 128 to 256 tokens, and then again at 512 tokens. The modest variance at 650 tokens is to be expected and can be explained by factors such as higher padding for shorter reviews, the inclusion of duplicated or noisy sections in very long reviews, and the increased complexity of optimizing over longer temporal sequences. Despite this minor variation, the overall pattern still favors longer sequence durations. This pattern is consistent with Yelp reviews, where sentiment-related information are frequently dispersed throughout the text. Shorter sequences miss key opinion information, while longer sequences enable the BiLSTM to capture a majority of sentiment trajectories.

In comparison, vocabulary size shows a weaker and more subtle influence on performance. The 20k and 30k vocabularies yield similar accuracies across all sequence lengths, suggesting both capture sufficient lexical diversity for the task. The 40k vocabulary, however, consistently underperforms due to sparsity and the presence of many rare, weakly trained token embeddings.

Overall, the sequence length is the most important representation hyperparameter, and a modest vocabulary size of 20k to 30k is appropriate. The best-performing configuration of 20k vocabulary words with a 650-token sequence length, which achieves the best test accuracy of approximately 67.62% is used as a baseline in subsequent model comparisons.

B. Model Hyperparameter Tuning

After determining that 20k vocabulary and 650-token maximum sequence length were the most effective representation configurations, we conducted a dedicated hyperparameter tuning experiment to find the best-performing baseline BiLSTM and CNN-BiLSTM model under these conditions.

We explored a well-defined hyperparameter search space for both architectures using Optuna. For the BiLSTM model, the search included embedding dimension $\{64, 128, 256\}$, hidden dimension $\{64, 128, 256\}$, number of LSTM layers $\{1, 2\}$, dropout $\{0.1, 0.2, 0.3, 0.5\}$, learning rate $\{5\times10^{-4}, 1\times10^{-3}, 2\times10^{-3}\}$, weight decay $\{0.0, 1\times10^{-5}, 1\times10^{-4}\}$, and batch size $\{32, 64, 128\}$. For the CNN-BiLSTM model, we used the same LSTM-related search space and additionally tuned convolution-specific parameters such as CNN output channels $\{64, 128, 192\}$, kernel size $\{3, 5, 7\}$, and number of CNN layers $\{1, 2\}$. In each Optuna trial, a unique combination of these hyperparameters was sampled, and the model was trained and evaluated for 3 epochs while monitoring validation loss and accuracy. The training and validation accuracies from all trials are summarized in Table 1.

The table shows some consistent trends. Moderate-capacity setups, particularly 128-dimensional embeddings with 64 hidden units and two LSTM layers, produced the highest validation accuracy for the BiLSTM model. Larger configurations, such as 256-dimensional embeddings or hidden sizes, did not increase performance and occasionally worsened validation accuracy, indicating slight overfitting. Similarly, dropout values around 0.2 seemed to strike a balance between regularization and model capacity, outperforming both lower and higher dropout values. We found that smaller convolutional modules (e.g., 64 output channels with kernel size 3) resulted in greater generalization than wider or deeper CNN stacks in the CNN-BiLSTM model.

After determining the ideal hyperparameter settings for each architecture, we fully trained the BiLSTM and CNN-BiLSTM models using these optimal configurations. The training and validation loss curves for both models are showcased in Figures 3 and 4, respectively.

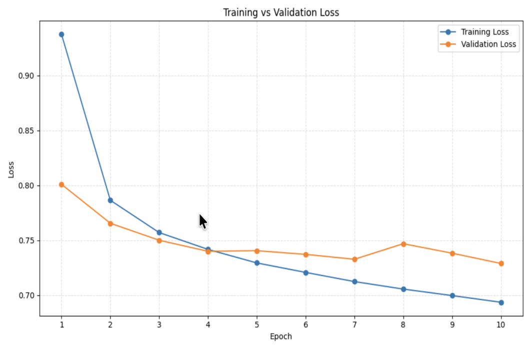


Fig. 3: Training v/s Validation Loss for BiLSTM Model

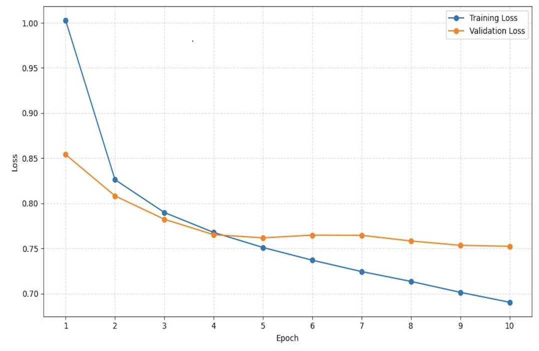


Fig. 4: Training v/s Validation Loss for CNN-BiLSTM Model

The training and validation loss curves for both models show that learning remained consistent throughout the training (10 epochs). The validation loss tracks the training loss very closely, which shows that both models are learning without overfitting. The BiLSTM's loss decreases gradually over time, while the CNN-BiLSTM drops more sharply at the beginning because the convolutional layers help capture useful local patterns early in training. Both models level off neatly, and the gap between their training and validation losses remains tiny. Overall, the curves suggest that the hyperparameters selected assist both models in learning and generalizing well.

C. Comparison between BiLSTM and CNN-BiLSTM Models

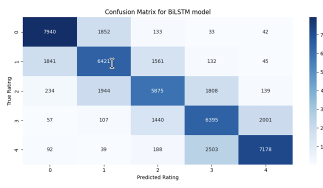
Table II shows that both models perform well for the extreme classes (ratings 0 and 4), while performance decreases for the middle ratings where user opinions are more ambiguous. For ratings 0 and 4, both models achieve strong precision and recall, with the BiLSTM performing slightly better. For the intermediate classes (ratings 1, 2, and 3), both models exhibit reduced accuracy and F1-scores, reflecting the difficulty of distinguishing these borderline reviews.

TABLE I: Hyperparameter Tuning Results for BiLSTM and CNN-BiLSTM (20k Vocab, 650 Max Seq Len)

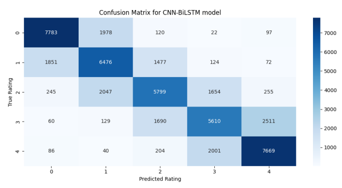
Model	Emb. Dim	Hid. Dim	L (Layers)	Drop Rate	LR	WD Decay	Batch Size	CNN Ch.	Kernel Size	CNN Layers	Train Acc	Val Acc
BiLSTM	256	256	1	0.3	5e-4	0	64	–	–	–	0.6775	0.6567
	64	256	1	0.3	2e-3	0	32	–	–	–	0.6697	0.6552
	128	**64**	**2**	**0.2**	**2e-3**	**1e-5**	**64**	–	–	–	**0.6714**	**0.6748**
	128	128	2	0.2	5e-4	1e-5	128	–	–	–	0.6543	0.6539
	64	128	1	0.1	5e-4	1e-5	128	–	–	–	0.6198	0.6295
CNN-BiLSTM	64	64	2	0.5	5e-4	1e-5	32	128	3	2	0.6506	0.6468
	64	64	1	0.1	2e-3	0	64	192	3	1	0.6553	0.6459
	256	256	1	0.5	2e-3	1e-4	32	192	7	2	0.6273	0.6338
	256	64	1	0.5	1e-3	0	64	64	3	2	0.6640	0.6530
	64	**128**	**2**	**0.3**	**5e-4**	**1e-5**	**32**	**64**	**3**	**1**	**0.6530**	**0.6541**

TABLE II: Comparison of BiLSTM and CNN-BiLSTM Model Performance Across Yelp Ratings

Class	Model	Precision	Recall	F1	Specificity	Acc./Class
Rating 0	LSTM	**0.7812**	**0.7940**	**0.7875**	**0.9444**	**0.9143**
	CNN-LSTM	0.7764	0.7783	0.7773	0.9440	0.9108
Rating 1	LSTM	**0.6196**	0.6421	**0.6307**	**0.9014**	**0.8496**
	CNN-LSTM	0.6069	**0.6476**	0.6266	0.8952	0.8456
Rating 2	LSTM	**0.6388**	**0.5875**	**0.6121**	**0.9170**	**0.8511**
	CNN-LSTM	0.6242	0.5799	0.6012	0.9127	0.8462
Rating 3	LSTM	0.5883	**0.6395**	**0.6128**	0.8881	**0.8384**
	CNN-LSTM	**0.5961**	0.5610	0.5780	**0.9050**	0.8362
Rating 4	LSTM	**0.7632**	0.7178	0.7398	**0.9443**	**0.8990**
	CNN-LSTM	0.7232	**0.7669**	**0.7444**	0.9266	0.8947



(a) BiLSTM



(b) CNN-BiLSTM

Fig. 5: Confusion matrices for (a) BiLSTM and (b) CNN–BiLSTM model

When comparing the two architectures, the BiLSTM model provides consistently better average precision, recall, and F1-score across the five rating categories. The CNN-BiLSTM model performs competitively. however, these results suggest that while the convolutional layer helps capture local phrase-level patterns, the BiLSTM alone is already effective at modeling the sequential dependencies in review text.

The BiLSTM and CNN–BiLSTM models attained an overall test accuracy of 67.62% and 66.68%, respectively. The confusion matrices in Fig. 5 of the two models show clear differences in how they handle each rating. For ratings 0 and 4, both models perform well, but the BiLSTM model produces slightly fewer mistakes. For ratings 1 and 2, the CNN-BiLSTM model reduces some misclassifications and spreads the errors more evenly across the neighboring classes. The results suggest that the BiLSTM model has a small advantage in direct class prediction for the middle ratings, but the CNN-BiLSTM model remains consistent across categories and avoids large classification jumps between distant ratings.

IV. Conclusion

The BiLSTM and CNN–BiLSTM hybrid architecture deep sequence modeling approaches were assessed in this study for fine-grained Yelp star rating prediction. We investigated how each method performs rating classification across polarized and ambiguous reviews under equal preprocessing and hyper-parameter settings.

With a test accuracy of 67.62%, the BiLSTM just beat CNN–BiLSTM's 66.68%. More significantly, the BiLSTM performed more consistently across evaluation criteria, especially for ratings 1-3, where F1-scores were significantly higher. The models did remarkably well on extreme ratings (0 and 4), when strong sentiment language provides clear categorization signals. Nearby classes (1-2 and 3-4), however, generated the most uncertainty, demonstrating how challenging it is still to identify minute sentiment gradations.

Convolutional feature extraction helped the CNN–BiLSTM learn more quickly at first, but this early advantage was short-lived. It achieved a plateau below BiLSTM performance by training completion, indicating that improved generalization on this task is not guaranteed by architectural complexity alone. Our data suggests that simpler models are the more sensible option in this case since they generalize more reliably and require less overhead. Stronger contextual modeling using attention processes or transformer-based techniques, which could better manage ambiguous intermediate ratings and lower borderline classification errors, is probably what will be needed for future advancements.

References

[1] G. Rao, W. Huang, Z. Feng, and Q. Cong, "LSTM with sentence representations for document-level sentiment classification," Neurocomputing, vol. 308, pp. 49–57, 2018.

[2] R. Belaroussi, S. C. Noufe, F. Dupin, and P.-O. Vandanjon, "Polarity of Yelp reviews: a BERT–LSTM comparative study," Big Data Cogn. Comput., vol. 9, no. 5, p. 140, 2025.

[3] B. He, L. Li, Y. Bo, and J. Zhou, "Bi-directional LSTM-GRU based time series forecasting approach," Int. J. Comput. Sci. Inf. Technol., vol. 3, no. 2, pp. 222–231, 2024.

[4] H. Khataei Maragheh, F. S. Gharehchopogh, K. Majidzadeh, and A. B. Sangar, "A hybrid model based on convolutional neural network and long short-term memory for multi-label text classification," Neural Process. Lett., vol. 56, p. 42, 2024.

[5] A. Alasmari, N. Farooqi, and Y. Alotaibi, "Sentiment analysis of pilgrims using CNN-LSTM deep learning approach," PeerJ Comput. Sci., vol. 10, p. e2584, 2024.