A Comparison of Recurrent Neural Networks and Pretrained Transformers for IMDb Sentiment Analysis

Jiayi Sun, Cindy Wang, Shuzhen Zhang
University of Washington
{sunjy025, cinnddyy, shuzhenz}@uw.edu

Abstract

Sentiment analysis of long form text remains challenging because sentiment cues are distributed across a document and involve negation, contrast, or late emerging opinion shifts. In this study, we compare recurrent neural networks (RNN) and pretrained Transformer models for binary sentiment classification on the IMDb movie review dataset. We evaluate a custom bidirectional LSTM classifier trained from scratch and finetuned pretrained Transformers, specifically GPT-2 and XLNet, using both built-in classification heads and custom multilayer perceptron (MLP) heads. To examine how truncation affects performance, we vary the maximum input sequence length across models. Our results show that pretrained Transformers outperform the LSTM baseline, with XLNet achieving the highest accuracy, particularly for longer sequences. This advantage remains consistent across head architectures and reflects XLNet's bidirectional context modeling via permutation based pretraining. Increasing the sequence length generally improves performance for all models, though gains diminish beyond 1024 tokens. While the LSTM offers computational efficiency and competitive performance given its simplicity, it struggles to capture very long range dependencies compared to attention based architectures. Overall, our findings highlight trade offs between accuracy, computational cost, and architectural flexibility in sentiment analysis of long documents, and demonstrate the effectiveness of pretrained Transformers for this task.

1 Introduction

IMDb movie reviews (1) provide a widely used benchmark for sentiment analysis because they contain natural, opinionated text with substantial variation in writing style and length. A key challenge in movie review sentiment classification is that sentiment evidence is often distributed throughout the document and can involve negation (e.g. "not good"), contrast (e.g. "great acting, but weak plot"), and shifts in opinion that appear late in the review. These properties make performance sensitive to preprocessing choices, especially tokenization and truncation, since models can only consume a limited number of tokens.

Recent progress in sentiment analysis has been driven by language models such as BERT (2), XLNet (3), and GPT-2 (4), as well as classical sequence models such as LSTM based RNNs (5). In this project, we compare two approaches: (i) a custom RNN classifier using an LSTM backbone, and (ii) finetuning pretrained Transformer models from HuggingFace, specifically **GPT-2** and **XLNet**, with an added task specific neural network head. We vary the maximum sequence length to study how truncation affects performance and to compare the pretrained models against the LSTM baseline on IMDb sentiment prediction.

2 Exploratory Data Analysis

We are working with movie reviews from the IMDb website. The training set contains 25,000 labeled reviews with binary sentiment labels (0/1). We examine label balance, review length, representative vocabulary patterns, and raw text artifacts

2.1 Class distribution

Figure 1 shows the label counts (label 0 negative: 12500, label 1 positive: 12500). The distribution is balanced, so we do not use class weighting or resampling.

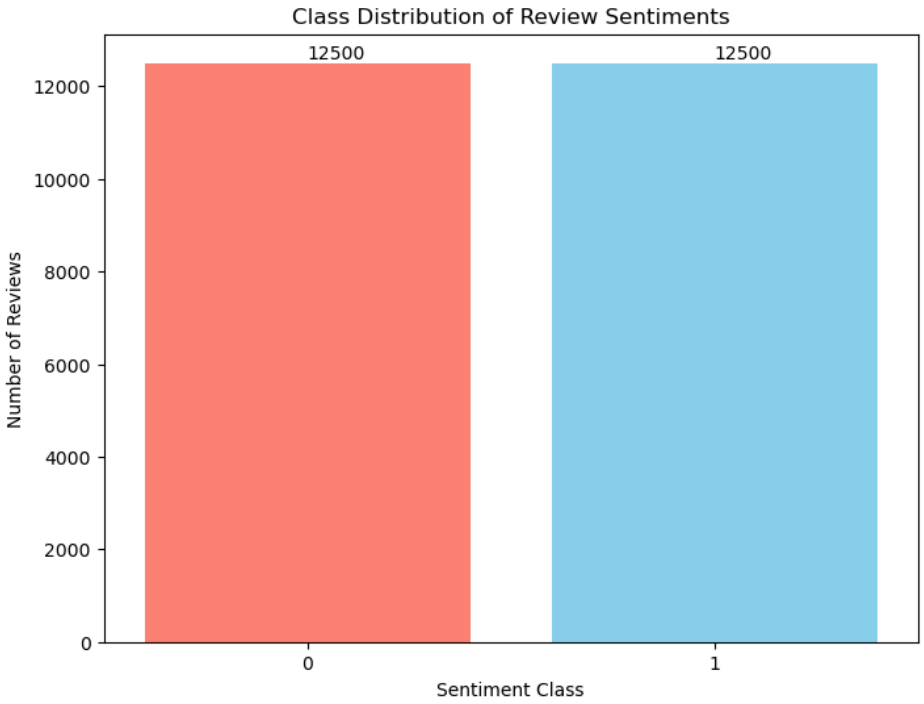
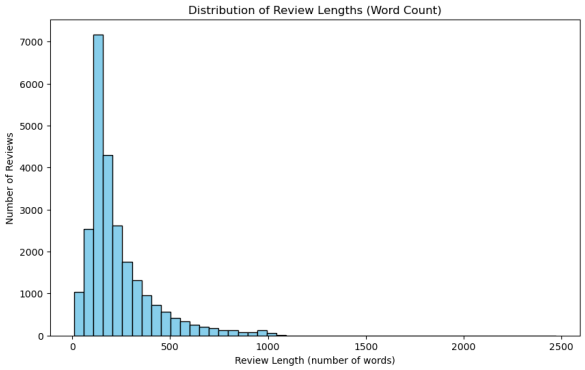


Figure 1: Distribution of review sentiments

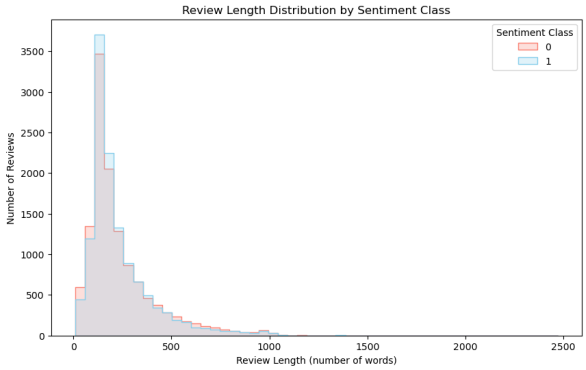
2.2 Review length

We examine review length to understand input variability and to guide truncation/padding decisions. We summarize length using word count.

- Typical length: mean 234, median 174
- Spread: 25th–75th percentile 127-284
- Long reviews: max 2470



(a) Review Length Distribution



(b) Review Length Distribution by Class

Figure 2

Figure 2a displays the review length distribution and shows a right skewed pattern. We also compare length distributions across labels in figure 2b. The two classes have similar typical lengths, suggesting length is unlikely to be a strong confounder.

2.3 Vocabulary and text content

We inspect common words to understand what content is associated with each sentiment class. To focus on meaningful terms, we report frequent words after basic normalization (e.g., lowercasing) and stopword removal.

After removing stopwords, the most frequent tokens are dominated by general movie-review language. The top terms include movie (30,887), film (27,780), one (22,515), and like (18,790), followed by evaluative but still broadly used words such as good (11,929), really (11,065), and great (7,719). This indicates that a large portion of the corpus discusses films in a generic way, while sentiment is often expressed through modifiers rather than topic words.

To examine how wording differs by sentiment class, we also computed "distinctive" words, tokens that appear much more often in one class than the other. For class 0, distinctive tokens include strongly negative descriptors such as unwatchable and awful, along with several proper nouns (e.g., seagal/segal, uwe, boll) that likely correspond to specific films or actors frequently mentioned in negative reviews. For class 1, distinctive tokens are also dominated by proper nouns (e.g., edie, gundam, antwone, visconti), suggesting that positive reviews may similarly cluster around specific titles, genres, or named entities.

One additional observation is that non-word artifacts such as "-" (9,644) and the token "*/" appearing among distinctive words suggest residual markup or formatting tokens remain in the text. This observation supports applying a light cleaning step (e.g., stripping HTML/markup and normalizing punctuation) before tokenization so that the vocabulary is not inflated by artifacts.

2.4 Qualitative inspection of sample reviews

We show 300 characters for readability.

Sample reviews for sentiment class 0:

- Sarah Silverman is really the "flavor of the month" comic right now. Is she really worth all the hype? Yes and no. She is funny at times, sometimes hilariously so (her standup routine is actually quite interesting, though not always funny). Other times, you're feeling cheated by the media for overhy
- *Spoiler warning*

First of all I rated this movie 2 out of 10.

The idea is good, but there are too many stupid errors in the movie, failing to make it the psyching drama that it might have been. First of all she never fights alone. After an initial very strange doubt from her m
- A group of people are invited to there high school reunion, but after they arrive they discover it to be a scam by an old classmate they played an almost fatal prank on. Now, he seeks to get revenge on all those that hurt him by sealing all the exits and cutting off all telephone lines.

D

In class 0, the sampled reviews contain clear negative judgments and dissatisfaction. Reviewers frequently point out flaws ("too many stupid errors"), emphasize low ratings ("2 out of 10"), and describe the film concept as wasted potential ("the idea is good, but…"). The tone often includes criticism of execution (plot, logic, acting) and sometimes uses informal intensifiers and sarcasm.

Sample reviews for sentiment class 1:

- Jess is 18, very smart and wants nothing more than to play football, when she joins a local team she has to lie to her parents again and again, as they would never approve of her chasing her dream, they want her to settle down with a nice Indian boy and learn how to cook.

Bend it Like Bec
- Pet Sematary is a very good horror film and believe it or not somebody can make a good horror film out of a Stephen King novel. Mary Lambert does a great job with this film and manages to bring across King's creepy story pretty well. Most people may avoid this, but they should check it out.
- This is an exquisite film about the search for some bliss in everyday life. The pacing, the camera work, the emotion, the haunting musical score and the pure charm of this picture make it a must see. It isn't easily appreciated by the immature or emotionally stunted. The only flaw I can see about th

In class 1, the sampled reviews show strong positive appraisal and recommendation language. Reviewers use explicitly positive descriptors ("very good," "exquisite," "must see"), highlight craftsmanship (pacing, camera work, emotion, musical score), and sometimes include a direct recommendation ("they should check it out"). These examples suggest that positive reviews are more likely to contain endorsement phrases and evaluative adjectives, while negative reviews often contain complaint framing and low-score statements.

2.5 Data artifacts and cleaning needs

We check for raw text artifacts that could mislead the model or inflate vocabulary size. We observe HTML tags in 58.67% of the total reviews, repeated punctuation in 36.49% of the reviews, and all cap words (4 or more letters) in 22.56 % of the reviews. These artifacts account for most of the noise in the dataset. Others artifacts include non-ASCII characters, URLs, email addresses.

Our cleaning steps include replacing HTML tags with spaces, normalizing white spaces, and removing any symbols that appear in reviews that are not related to sentiment. We would like to keep words in all caps and repeated punctuation such as '!!!' as these both carry sentiment intensity.

3 Methodology

3.1 RNN Model

We choose RNNs for the IMDb sentiment classification because they model sequential structure directly. Unlike bag-of-words methods that ignore word order, RNNs process tokens sequentially and maintain a hidden state that captures contextual information from preceding words, which is essential for understanding sentiment expressions involving negation or modification.

The basic architecture of the RNN is illustrated in 3. At time step t, the recurrent encoder updates its hidden state based on the current input embedding x_t, and the previous hidden state h_{t-1}

$$h_t = f(W_x x_t + W_h h_{t-1} + b) \tag{1}$$

where $f(\cdot)$ denotes a nonlinear activation function, this recurrence allows earlier tokens to influence later representations (5).

Furthermore, sentiment relevant cues may depend on both short-range and long-range dependencies within a sentence or document. RNN variants such as Long Short-Term Memory (LSTM) are specifically designed to mitigate vanishing gradient issues and effectively capture such dependencies. LSTMs introduce a cell state c_t and gating mechanisms that control information flow across time steps, the update follows

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f), \tag{2}$$
$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i), \tag{3}$$
$$\tilde{c}_t = \tanh(W_c[h_{t-1}, x_t] + b_c) \tag{4}$$
$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \tag{5}$$
$$h_t = o_t \odot \tanh(c_t) \tag{6}$$

which allow the model to retain or discard information as needed across long sequences

Finally, given the moderate size and typical sequence length of the IMDb dataset, RNN-based models provide an appropriate balance between representational power and computational efficiency, making them a practical and effective choice for this task.

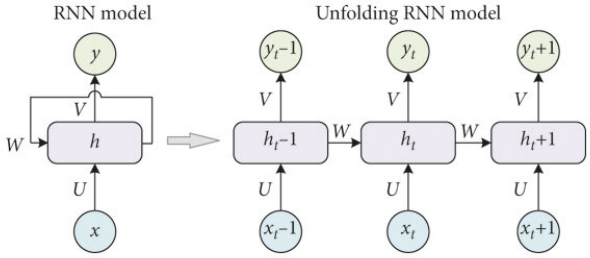


Figure 3: Basic Structure of RNN

3.1.1 RNN Construction and Fine-tuning

Configuration file: We define all experiment settings in a single hyperparameters block at the start of the training notebook, including vocabulary size, maximum sequence length, embedding dimension, hidden state size, batch size, learning rate, number of training epochs, and device specification 3.

Tokenizer: We implement a corpus-driven tokenizer tailored to the IMDb dataset. Starting from raw reviews, the tokenizer applies deterministic text normalization, including lowercasing and basic punctuation handling, and segments each document into word level tokens. We construct a task-specific vocabulary from the training split by counting token frequencies and retaining the most frequent tokens under a fixed capacity. We map rare or unseen tokens to an `<UNK>` symbol. Each review is then represented as an ordered sequence of integer token IDs, which preserves word order and contextual dependencies relevant to sentiment prediction.

Preprocessing: We clean raw IMDb reviews based on deterministic rules that replace non-breaking spaces, remove HTML tags, and normalize whitespace. We then tokenize the normalized text using a word-level tokenizer that preserves linguistically salient patterns, including apostrophe containing words, repeated punctuation, ellipses, and single character symbols. We encode fully capitalized words with an explicit <ALLCAPS> feature token to capture emphasis. We process each review in `TextDataset.__getitem__`, where we truncate token sequences to a fixed maximum length and convert them to integer indices via `Tokenizer.encode`. For mini-batch training, variable-length sequences are dynamically padded with `pad_sequence`, and their true lengths are retained and passed to the recurrent encoder through `pack_padded_sequence`.

Model: The model consists of an embedding layer followed by a bidirectional LSTM encoder. The LSTM processes each review sequentially and produces hidden states that summarize contextual information from both directions. We construct a sequence-level representation by concatenating the final forward and backward hidden states and pass this representation to a linear classification layer that outputs a single sentiment logit.

Training loop: We train the model end to end with mini-batch gradient descent. We optimize binary cross-entropy loss with logits with an adaptive optimizer and update parameters once per batch.

Evaluation and model selection: At the end of each epoch, we evaluate the model on the validation set and compute `{accuracy, precision, recall, F1 score}`. We save model checkpoints and use validation performance to compare runs.

3.2 GPT2 and XLNET

We choose GPT-2 and XLNet because they represent two influential but contrasting pretraining strategies for Transformer language models. GPT-2 uses a left to right autoregressive objective that predicts each token based on all previous tokens in the sequence. Given a token sequence $x = (x_1, ..., x_T)$ (4),

$$\max_{\theta} \mathbb{E}_{z \sim \mathcal{Z}_T} \left[\sum_{t=1}^{T} \log p_\theta(x_{z_t} | x_{z_{<t}}) \right] \tag{7}$$

This objective performs well for language modeling and text generation, and GPT style models dominate many modern NLP systems. For these reasons, GPT-2 provides a strong baseline for sentiment classification.

XLNet uses a permutation based language modeling objective that conditions on tokens from both left and right context without masking. By training over many factorization orders of the same sequence, XLNet captures bidirectional dependencies while avoiding the pretrain finetune mismatch introduced by masked language models. Let $z = (z_1, ..., z_T)$ denote a permutation of $\{1, ...T\}$ and let $\mathcal{Z}_T$ be the set of all such permutations (3),

$$\max_{\theta} \sum_{t=1}^{T} \log p_\theta(x_t | x_{<t}) \tag{8}$$

This design works well for long document sentiment analysis, where sentiment cues may appear anywhere in the review and may depend on distant word interactions.

When we evaluate GPT-2 and XLNet on the same IMDb task, we directly compare how their pretraining objectives and context usage affect performance.. We vary the maximum input length to study truncation effects and to test whether these architectural differences lead to consistent performance gaps on long movie reviews.

To adapt pretrained language models to our binary sentiment classification task, we built a fine-tuning pipeline based on the HuggingFace ecosystem that (i) loads a pretrained tokenizer and model, (ii) tokenizes and dynamically pads text

batches, and (iii) trains the classification head end-to-end on our labeled dataset. We experimented in two stages: (1) an initial baseline using pretrained models defined on Huggingface and (2) an extension that extracts the pretrained models last hidden states, pools them into a sequence-level representation, and feeds it into a custom MLP head trained with cross-entropy loss.

3.2.1 Stage 1: Fine-tuning with the built-in classification head (baseline)

Configuration file: We store key hyperparameters in `config.json` (e.g., maximum sequence length, learning rate, weight decay, batch sizes, gradient accumulation steps, warmup ratio, and number of epochs 4). This makes ablation studies reproducible and easy to run by editing a single file.

Tokenizer: We load the pretrained tokenizer associated with each checkpoint. For GPT2 family models (which often do not define a padding token), we set `pad_token` to `eos_token` to enable correct batching and padding. Because each pretrained model uses its own fixed vocabulary, we do not construct a custom tokenizer.

Preprocessing: We tokenize raw text with truncation to `max_length`. Padding is performed dynamically at batch time using a collator, which avoids padding all samples to a global fixed length and typically improves efficiency.

Model: We start from the pretrained checkpoint and load the standard `*ForSequenceClassification` model (e.g., `GPT2ForSequenceClassification` for GPT2-family), which attaches a trainable classification head to the Transformer backbone.

Training loop: We fine-tune with `Trainer`, enable gradient accumulation to handle memory constraints, warmup scheduling for early stability, and save model checkpoints at the end of each epoch.

Evaluation and model selection: We train the model by minimizing cross-entropy loss. We report `{accuracy, precision, recall, F1 score}` as the main performance metric. We compute validation **log loss** to select the best checkpoint during training. At the end of training, we reload the checkpoint with the lowest validation log loss for final evaluation.

3.2.2 Stage 2: Adding a custom MLP head

After establishing the baseline, we finetune the pretrained backbone with a custom MLP classification head.

Feature extraction from the pre-trained model.

The pre-trained model returns the final hidden states `last_hidden_state` with shape `[batch, seq_len, hidden_dim]`. To obtain a single representation per input sequence, we select the hidden state of the **last valid token** using the attention mask. This pooling strategy produces a sequence-level embedding that preserves information from the full truncated review.

MLP head architecture.

Let $\mathbf{h} \in \mathbb{R}^d$ denote the pooled hidden representation (where d is the pre-trained model hidden size). The classifier is a lightweight two-layer MLP:

$$\mathbf{z} = W_2\big(\mathrm{Dropout}(\mathrm{GELU}(W_1\mathbf{h} + b_1))\big) + b_2,$$

where $\mathbf{z}$ are the output logits for binary classification. This design adds capacity beyond a single linear head while remaining small enough to train efficiently.

Trainer compatibility and checkpointing.

The forward pass returns a standard `SequenceClassifierOutput` containing `loss` (cross-entropy when labels are provided) and `logits`, so it can be trained with the same `Trainer` setup as the baseline. We also implemented `save_pretrained` and `from_pretrained` so that both the pre-trained model and the MLP head can be saved and reloaded consistently. We also support optional gradient checkpointing to reduce memory usage during training.

4 Results and Summary

4.1 RNN Model Result

Table 1 reports the results of the RNN with the customized tokenizer encoding under different hyperparameter settings.

Table 1: Result Metrics for RNN Model

Model	max_len	epochs	Accuracy	Precision	Recall	F1	Val Loss	Test Accuracy[1]
RNN	1024	6	0.8932	0.8885	0.8992	0.8938	0.2904	0.8905
	500	7	0.8828	0.9086	0.8512	0.8790	0.3118	0.8727

[1] Test Accuracy is computed on Kaggle (1) using 50% of the test set.

4.2 Pretrained Model Result

Table 2 reports the results of both stages: the baseline (built-in head) and the extended model (with MLP head), under different hyperparameter settings.

Table 2: Comparison of built-in classification head vs. our custom last-layer + MLP head.

Model	max_len	epochs	Accuracy	Precision	Recall	F1	Val Loss	Test Accuracy[1]
DistilGPT2	1024	3	0.9172	0.9046	0.9328	0.9184	0.2355	0.9282
	500	3	0.9136	0.9014	0.9288	0.9149	0.2460	0.9225
DistilGPT2 + MLP	1024	3	0.9240	0.9193	0.9296	0.9244	0.3470	0.9299
	500	3	0.9156	0.9043	0.9296	0.9168	0.3267	0.9243
XLNet	1200	3	0.9468	0.9436	0.9504	0.9451	0.3347	0.9571
	1024	3	0.9468	0.9436	0.9504	0.9470	0.3083	**0.9578**
	500	3	0.9384	0.9412	0.9352	0.9382	0.3503	0.9524
XLNet + MLP	1200	3	0.9452	0.9470	0.9432	0.9451	0.3515	0.9570
	1024	3	0.9456	0.9470	0.9440	0.9455	0.3253	**0.9572**
	500	3	0.9384	0.9370	0.9400	0.9385	0.3723	0.9525

[1] Test Accuracy is computed on Kaggle (1) using 50% of the test set.

5 Discussion

5.1 Model Comparison

We focus primarily on accuracy because the Kaggle leaderboard evaluates submissions using this metric. Accuracy measures the proportion of correctly classified reviews, defined as

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{9}$$

where TP (True Positive) denotes the number of samples that are correctly predicted as positive, TN (True Negative) denotes the number of samples that are correctly predicted as negative, FP (False Positive) denotes the number of samples that are incorrectly predicted as positive, and FN (False Negative) denotes the number of samples that are incorrectly predicted as negative.

Across all Transformer based experiments, XLNet consistently achieves higher accuracy than GPT-2. This pattern holds across all sequence lengths and for both the built-in classification head and the custom MLP head. The result reflects differences in pretraining objectives: XLNet's permutation based language modeling incorporates bidirectional context during pretraining, which appears particularly effective for long form sentiment analysis where sentiment cues may occur late in the review or depend on distant context.

Adding a custom MLP head produces small but consistent improvements for GPT-2 across multiple sequence lengths. This result suggests that GPT-2's pretrained representations benefit from a slightly more expressive classifier that can better adapt token level features to the sentiment task. In contrast, the improvements from the MLP head are less pronounced for XLNet, indicating that its built-in classification head already extracts strong sequence level representations. Overall, these findings suggest that classifier design can matter for pretrained models, but the impact depends on the strength and structure of the underlying backbone.

For both GPT-2 and XLNet, increasing the maximum sequence length generally improves accuracy, as longer inputs preserve more sentiment relevant information and reduce truncation effects. However, the gains diminish at larger lengths. For XLNet, the 1024 token setting slightly outperforms the 1200 token setting, suggesting that very long sequences may introduce noise or limited additional signal relative to their computational cost.

The RNN model achieves lower accuracy than the Transformer based models but remains competitive given its simpler architecture. With a maximum sequence length of 1024, the bidirectional LSTM reaches an accuracy of 0.8932, compared to 0.8832 at length 500. This improvement mirrors the trend observed in Transformers and confirms that longer context benefits sentiment classification even for recurrent models. However, the performance gap indicates that sequential recurrence alone struggles to capture very long range dependencies as effectively as attention based architectures.

From a computational perspective, the models exhibit clear trade offs. GPT-2 and the RNN both train in approximately 30 minutes under comparable settings. The RNN converges in 6–7 epochs due to early stopping, which limits training cost despite using longer sequences. XLNet incurs substantially higher computational cost, with training times around one hour, especially at larger sequence lengths. Increasing maximum sequence length consistently increases training time for all models.

Overall, the results highlight a balance between accuracy and efficiency. XLNet provides the strongest performance on long text sentiment classification but requires greater computational resources. GPT-2 offers a strong compromise between accuracy and training cost, while the RNN provides the simplest and fastest converging model at the expense of lower accuracy. Precision, recall, and F1 score closely track accuracy across all settings, indicating balanced performance differences rather than shifts in error trade offs.

5.2 Limitations and Future Work

The RNN model suffers from inherent limitations due to its sequential computation. Although the bidirectional LSTM captures local and moderate range dependencies, it processes tokens step by step and struggles to model very long range context efficiently, which leads to weaker performance on long reviews. The pretrained Transformer models introduce a different limitation. Frameworks such as HuggingFace provide pretrained models that are easy to load and finetune, which greatly simplifies experimentation, but this convenience reduces flexibility. Making substantial architectural changes to pretrained backbones, such as modifying attention mechanisms or internal layer structure, requires significant engineering effort and limits the scope of modifications explored in this study. These constraints reflect the trade offs observed in the model comparison between accuracy, computational cost, and architectural flexibility.

Future work could explore a broader range of modeling choices. On the classification side, different custom heads could be evaluated, including deeper MLPs, attention based pooling layers, or hybrid heads that combine recurrent and feedforward components. On the backbone side, additional pretrained models such as DeBERTa could be investigated, as its disentangled attention mechanism may further improve long text sentiment modeling. Finally, training a language model from scratch using a more rigorous pretraining setup, such as a smaller Transformer or recurrent language model trained on a large corpus, could provide greater flexibility and deeper insight into how pretraining objectives affect downstream sentiment performance, although at a higher computational cost.

6 Contributions of the Group Members

Shuzhen Zhang: Completed the implementation of the DistilGPT2 and XLNet training pipelines. Designed and integrated a custom MLP head into the original model architecture.

Cindy Wang: Completed the EDA section, wrote up majority of the report, ran all the code and provided results.

Jiayi Sun: Collaboratively contributed to the implementation of the RNN model and the preparation of the project report.

A Hyperparameters for RNN

Parameter	Value
Validation fraction	0.1
Min token frequency	2
Max sequence length	500/1024
Embedding dim	300
Hidden dim	300
LSTM layer	2
Dropout	0.5
Batch Size	16
Max epochs	10
Loss function	BCEWithLogitsLoss
Weight decay	1e-4
Optimizer	Adam (lr = 1e-3)

Table 3: RNN Hyperparameters

B Hyperparameters for GPT-2 and XLNet

Parameter	Value
Model names	xlnet-base-cased/DistilGPT2
Validation fraction	0.1
Max sequence length	500/1024/1200
Epochs	3
Train batch size	4
Eval batch size	8
Grad accumulation steps	8
MLP: hidden dim	256
MLP: dropout	0.1
MLP: loss function	CrossEntropyLoss

Table 4: GPT-2 and XLNet Hyperparameters

References

[1] ArmeenTaeb and Xiaozhu Zhang. Win26 stat 528 kaggle competition 1. https://kaggle.com/competitions/win-26-stat-528-kaggle-competition-1, 2026. Kaggle.

[2] Kavita Arora, Neha Gupta, and Sonal Pathak. Sentimental analysis on imdb movies review using bert. In *2023 4th International Conference on Electronics and Sustainable Communication Systems (ICESC)*, pages 866–871, 2023. doi:10.1109/ICESC57686.2023.10193688.

[3] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. Xlnet: Generalized autoregressive pretraining for language understanding, 2020. URL: https://arxiv.org/abs/1906.08237, arXiv:1906.08237.

[4] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019. URL: https://api.semanticscholar.org/CorpusID:160025533.

[5] U.B. Mahadevaswamy and P. Swathi. Sentiment analysis using bidirectional lstm network. *Procedia Computer Science*, 218:45–56, 2023. URL: https://www.sciencedirect.com/science/article/pii/S1877050922024930, doi:10.1016/j.procs.2022.12.400.