Sentiment Analysis for Chapter Rating Prediction

GitHub Repository: https://github.com/rsm-jic147/MGTA_415_Project

Boyu Wang
University of California San Diego
bow019@ucsd.edu

Jiayin Chen
University of California San Diego
jic147@ucsd.edu

1 Introduction

In today's novel market, many authors have transitioned from traditional full-book publications to serialized releases, updating their stories daily or weekly. This shift allows continuous reader engagement and real-time narrative adaptation. However, a key challenge remains: while readers often comment on individual chapters, they rarely assign ratings, making it difficult for authors to gauge satisfaction accurately and risking audience loss if unfavorable content persists.

To address this, sentiment analysis can serve as a proxy for chapter-specific ratings, enabling authors to identify and adjust content that may not resonate with their audience, thereby improving reader retention. This approach fosters a responsive storytelling process and aligns with the dynamic nature of modern serialized fiction. Platforms like Royal Road and Wattpad have transformed reading into a shared experience, where readers anticipate new chapters and actively participate in discussions, strengthening engagement.[1] Serialized storytelling thrives on suspense and community interaction, encouraging speculation and fostering online communities, much like the anticipation of weekly television shows.[2]

However, most serialized platforms lack a chapter-specific rating system, as requiring frequent ratings would discourage engagement. Consequently, predicting chapter ratings from textual comments has become a significant challenge. Incorporating sentiment analysis allows authors to leverage reader feedback effectively, gaining insights into audience emotions and preferences. This data-driven approach enhances reader satisfaction and loyalty.[3]

In conclusion, integrating sentiment analysis into serialized fiction provides authors with valuable insights to refine their narratives. By actively monitoring and responding to feedback, authors can cultivate a loyal readership and enhance engagement in the competitive landscape of modern literature.

2 Hypothesis

We hypothesize that although differences in reading habits and publishing practices have brought significant transformations to the novel market, there are underlying similarities between full-book reviews and chapter-specific comments. While serialized updates encourage more frequent engagement, we believe that the fundamental way users express their opinions about a novel remains consistent across both full-book reviews and individual chapter comments. Given this assumption, we aim to leverage past user reviews and ratings of entire books to map these patterns onto chapter-specific evaluations. By analyzing historical full-book ratings and their associated textual reviews, we propose a method to predict chapter-level scores based on user comments, thus generating an estimated rating per chapter. This approach could provide valuable insights for authors, allowing them to track reader satisfaction in real time and make data-driven adjustments to their storytelling strategies.

3 Methodology

To validate our hypothesis, we utilize the Amazon Book Reviews Dataset as our training set.[4] This dataset contains a vast collection of user reviews and ratings for books, providing a rich resource for understanding how users express their opinions on complete works. The structured nature of

[1] https://newbookrecommendation.com/the-impact-of-serialized-books-on-reader-engagement-statistics

[2] https://horrortree.com/the-rise-of-serialized-fiction-and-its-impact-on-readers

[3] https://contentsquare.com/guides/sentiment-analysis/examples

[4] https://amazon-reviews-2023.github.io/

these reviews, which include both textual feedback and numerical ratings, makes them ideal for training a predictive model that maps review sentiment to rating scores. In addition to this, we collected chapter-specific comments from Wuxiaworld.com, a popular platform known for serialized web novels. Unlike Amazon reviews, which typically evaluate an entire book, Wuxiaworld comments are more dynamic, often reflecting reader reactions to individual chapters. These comments, however, lack direct numerical ratings, making it a suitable test case for our model's ability to predict chapter-level sentiment and scores. Our goal is to train a machine learning model on Amazon book reviews, learning the relationship between textual content and ratings, and then apply this model to predict chapter ratings based on Wuxiaworld comments. If successful, this would demonstrate that patterns in full-book reviews can be leveraged to estimate ratings for serialized chapters, validating our hypothesis.

Data Collection

- Extract book reviews and corresponding ratings from the Amazon Books dataset.
- Gather chapter-specific comments from Wuxiaworld.com as the test set.

Preprocessing & Feature Engineering

- Clean and tokenize textual data, removing noise such as stopwords and punctuation.
- Convert text into numerical representations using TF-IDF, word embeddings (Word2Vec, BERT, etc.), or sentiment scores.

Model Training

- Train a regression or classification model using Amazon book reviews, predicting ratings based on textual features.
- Experiment with different algorithms such as linear regression, random forests, and deep learning models (BERT, LSTM, etc.).

Testing on Wuxiaworld Comments

- Apply the trained model to predict ratings for chapter comments on Wuxiaworld.
- Evaluate whether the predicted ratings align with expected reader sentiment and engagement patterns.

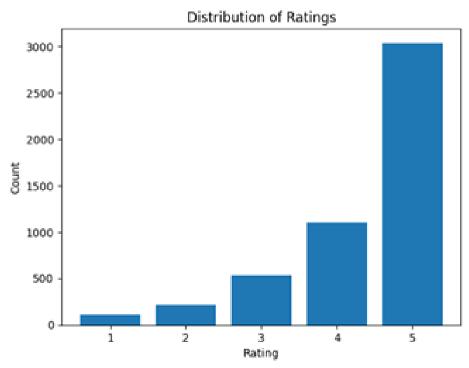


Figure 1: Distribution of Ratings

4 Exploratory Data Analysis

4.1 Overview of Dataset

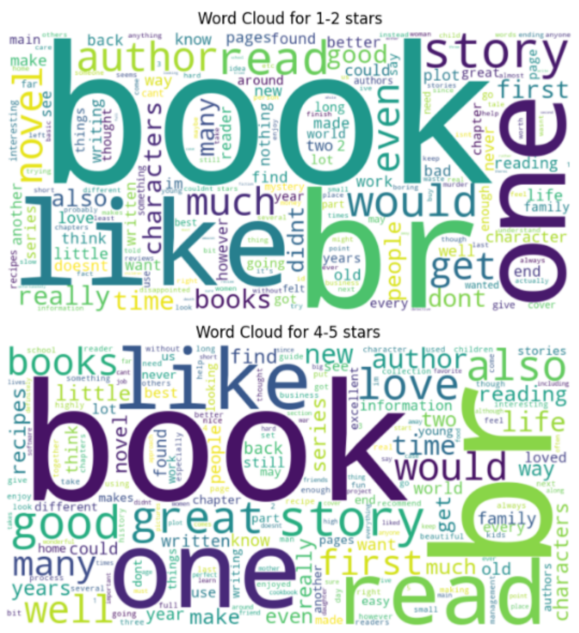
The Amazon Books dataset consists of a total of 10.3 million reviews, making it one of the largest publicly available book review datasets. Due to its vast size, we selected a 20,000-sample subset for our training and validation set to conduct our initial experiments efficiently. If necessary, we can later extend our study to incorporate the full dataset to further improve model performance and robustness.

4.2 Distribution of Ratings

The distribution of ratings in our selected subset, as shown in the figure, exhibits a strong skew toward higher ratings. The majority of reviews fall within the 4-star and 5-star categories, with 5-star ratings being the most frequent by a significant margin. In contrast, low ratings (1-star and 2-star reviews) are much less common, suggesting that most books in this dataset receive generally positive feedback.

4.3 Word Frequency in Reviews for Each Rating

To analyze the differences in user sentiment across different rating levels, we generated word clouds for each rating category, highlighting the most frequently appearing words in reviews. This visualization provides valuable insights into the language patterns associated with highly rated and poorly rated books.



From the high-rating categories (4-star and 5-star reviews), we observe that the most prominent words tend to be overwhelmingly positive. Words such as "great," "love," "amazing," and others reflect strong approval and appreciation. Additionally, the frequent occurrence of terms like "book," "author," "story," "series," and "characters" suggests that readers particularly value the overall narrative quality and writing style.

In contrast, low-rating categories (1-star and 2-star reviews) contain a higher concentration of negative terms, indicating dissatisfaction with various aspects such as plot execution, pacing, or character development. Words like "bad," "boring," "waste," and others highlight the frustration and discontent of reviewers. Furthermore, the presence of words such as "money," "buy," and "waste" suggests that some users felt their purchase was not worth the price.

This textual sentiment analysis reinforces our hypothesis that word choice in reviews strongly correlates with rating scores. By leveraging these patterns, our model can better predict ratings from textual comments on platforms like Wuxiaworld, where chapter reviews lack direct numerical scores.

5 Sentiment Analysis

To analyze the sentiment distribution of reviews across different rating levels, we used TextBlob for sentiment polarity detection, where scores range from -1 (negative) to +1 (positive). The density plot shows a clear distinction between rating groups: lower ratings (1-star and 2-star) are concentrated in the negative polarity range, while higher ratings (4-star and 5-star) shift toward positive sentiment, aligning with expected user satisfaction trends. Meanwhile, 3-star ratings are centered around neutral sentiment, reflecting mixed or lukewarm reviews. The strong correlation between textual sentiment and numerical ratings suggests that sentiment analysis can effectively infer chapter ratings for platforms like Wuxiaworld, where explicit scores are absent. Leveraging sentiment-based predictive modeling could further enhance rating predictions and provide valuable insights for authors and readers.

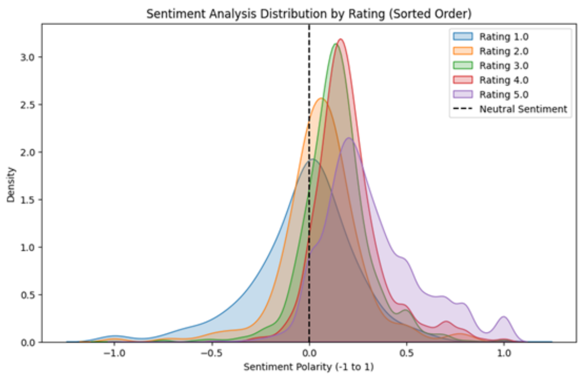


Figure 2: Sentiment Analysis Distribution

6 Predictive Task

Our objective is to train a predictive model using the Amazon Books dataset to predict the rating of a book based on its textual review. Given that book reviews contain rich sentiment and opinion information, we aim to develop a model that can accurately map textual features to numerical ratings (1 to 5 stars).

6.1 Text pre-processing

In the Text Pre-processing phase, we experiment with two distinct pipelines—one using NLTK and the other using spaCy—to transform raw review text into a cleaner format suitable for downstream modeling.

6.1.1 NLTK-Based Processing

NLTK (Natural Language Toolkit) is a well-established Python library widely used for research and educational purposes in the field of natural language processing. It provides a variety of tokenizers, stemmers, and other linguistic tools. In our pipeline, we split each review into sentences and then into individual words, using Porter Stemming to reduce words to their base stems. For example, "enjoyed" and "enjoying" would both map to "enjoy." This approach can effectively reduce vocabulary size, though it often discards certain linguistic nuances (e.g., tense, plurality).

6.1.2 spaCy-Based Processing

spaCy is a more production-focused library known for its efficient, modern NLP pipeline and advanced features like named-entity recognition, part-of-speech tagging, and dependency parsing. Unlike stemming, spaCy applies lemmatization, converting words to their dictionary forms—for instance, "running," "ran," and "runs" would all become "run," preserving more semantic clarity than basic stemming. Additionally, spaCy handles sentence segmentation and tokenization automatically with its built-in language models.

In both approaches, we also have the option to remove stopwords—high-frequency words like "the" or "and" that typically carry limited semantic value. Finally, the resulting tokens (or lemmas) can be joined back into strings or directly converted into numeric vectors (e.g., TF-IDF) for training our machine learning and deep learning models.

6.2 Vectorizer

In our experiment, we tested two methods for converting text into feature vectors: CountVectorizer and TfidfVectorizer, then compared how they performed in Linear Regression and Random Forest models.

6.2.1 CountVectorizer

- **Concept:** This method counts how many times each word appears in each document and produces a sparse matrix whose dimensions generally correspond to the total number of distinct words in the corpus. For instance, if a particular word appears three times in a review, its corresponding entry in that document's feature vector would be "3."

- **Characteristics:** It is simple, intuitive, and quickly quantifies text into learnable features. However, because high-frequency words (like "very" or "good") often receive higher counts, the resulting feature set may include noise rather than offering truly discriminative information.

6.2.2 TfidfVectorizer

- **Concept:** Building upon CountVectorizer, it incorporates Inverse Document Frequency (IDF) information to produce TF-IDF (Term Frequency–Inverse Document Frequency) features. TF-IDF adjusts a word's weight based on how common or rare it is across the entire document set:
  - If a word is frequent in the current document but not commonly seen in the overall corpus, it will receive a higher TF and IDF, thus a higher weight in the final vector.
  - In contrast, if a word appears in many documents, its IDF is relatively low, reducing its weight accordingly.

- **Characteristics:** By minimizing the influence of extremely common words, TF-IDF highlights more distinctive words that help differentiate documents. This often leads to better performance in classification or regression tasks.

In our code, we start by cleaning and tokenizing the raw text by using both NLTK and spaCy, then apply either CountVectorizer or TfidfVectorizer to transform the tokens into numeric vectors. These vectors are subsequently fed into models like Linear Regression and Random Forest to predict review ratings, with performance measured by metrics such as MSE. Our tests indicate that TfidfVectorizer often proves more effective in emphasizing words that carry unique meaning, making it generally more suitable in scenarios involving a variety of text types or tasks requiring a high level of sentiment understanding.

Model	Pre-processing	Vectorizer	MSE
Linear Regression	NLTK	Count	126.83
	spaCy	Count	118.98
	NLTK	TF-IDF	4.08
	spaCy	TF-IDF	6.41
Random Forest	NLTK	Count	0.73
	spaCy	Count	0.68
	NLTK	TF-IDF	0.71
	spaCy	TF-IDF	0.66

Table 1: Model performance comparison (MSE scores)

It is evident that the choice of vectorization method is crucial to model performance. In the case of linear regression, using CountVectorizer leads to an extremely high MSE (over one hundred), whereas switching to TF-IDF drastically reduces the MSE to around single digits. Furthermore, the choice of model also has a notable impact: under both Count and TF-IDF—regardless of whether NLTK or spaCy is used—Random Forest clearly outperforms linear regression (with MSEs around 0.6–0.7). Looking more closely at different preprocessing approaches, while the differences are not as pronounced as those from vectorization or model selection, the combination of spaCy + TF-IDF in Random Forest yields the best score (0.66). On the other hand, for linear regression, NLTK + TF-IDF slightly outperforms spaCy + TF-IDF. Overall, Random Forest is more robust in general, and TF-IDF tends to deliver more discriminative features than CountVectorizer in most scenarios—especially where higher predictive accuracy or nuanced language understanding is required.

6.3 Model

We begin with an extremely simple yet essential baseline: always predicting the average rating of the training set. Concretely, we calculate the mean of all the ratings in the training data and, during testing, we make that same mean rating the prediction for any given review text. The primary reason for introducing this seemingly "crude" method is to establish a reference benchmark for more complex models that follow: if the trained model cannot substantially surpass this average-based prediction, it indicates that the model is not effectively leveraging the key information in the review text. This simple Mean Rating baseline helps us quickly assess the impact of subsequent algorithmic improvements, and it also directly reflects the overall rating distribution in the reviews.

6.3.1 Random Forest

A Random Forest is an ensemble learning method based on multiple decision trees. Each tree in the forest makes a prediction, and these individual predictions are then averaged (in a regression task) to produce a final rating estimate. By training numerous trees on different subsets or features of the data, Random Forests reduce the variance of single decision-tree models and are generally better at capturing complex interactions among input features. In our code, after converting the textual data into numeric vectors (for instance, using TF-IDF), we feed these vectors and their corresponding ratings to a RandomForestRegressor. The model is trained to minimize Mean Squared Error (MSE), which encourages it to accurately match predicted ratings to the actual user ratings in the training set. Random Forests are relatively fast to train compared to deep learning methods and can handle high-dimensional feature spaces, making them a strong, non-linear baseline model. However, while they often outperform simpler approaches such as linear regression, they do not inherently capture the sequential order of words and may require large numbers of trees for stable results, especially when working with extensive vocabularies. Nonetheless, Random Forest remains a robust, interpretable starting point before shifting to more computationally intensive solutions like LSTM or BERT.

Optimization: To optimize our LSTM model, we experimented with different dropout rates, the number of LSTM layers, and hidden layer sizes to balance performance and generalization. We tuned dropout values (e.g., 0.01) to prevent overfitting while preserving important temporal dependencies. Additionally, we tested varying LSTM layer sizes (e.g., 64 and 32 units) and added a dense hidden layer with 16 neurons to improve feature extraction. To avoid excessive training and minimize overfitting, we applied early exit (early stopping), which monitors validation loss and stops training if no improvement is observed for a set number of epochs. The figures show that early stopping was triggered when the validation error started increasing, preventing further overfitting.

6.3.2 LSTM

Long Short-Term Memory (LSTM) networks are a specialized type of Recurrent Neural Network (RNN) designed to handle long-range dependencies in sequence data, such as the natural language sequences found in user reviews. Unlike standard feedforward networks, LSTMs contain memory cells and gating mechanisms—commonly referred to as input, forget, and output gates—which control how information is retained, updated, or removed across time steps. This architecture helps overcome problems like vanishing or exploding gradients, allowing the model to capture important context that appears early or late in a sequence.

In our implementation, we first transform raw text into integer sequences (token IDs), then apply padding or truncation to ensure uniform input lengths. The model itself typically includes an embedding layer (mapping each token ID to a dense vector), an LSTM layer for capturing temporal dependencies, and a final dense layer that outputs a single numeric rating. We employ the Adam optimizer and use Mean Squared Error (MSE) as our loss function to align with the regression nature of rating prediction. Between layers, dropout is introduced to reduce overfitting, and early stopping is used to prevent unnecessary epochs once validation performance plateaus.

Optimization: To optimize our LSTM model, we experimented with different dropout rates, the number of LSTM layers, and hidden layer sizes to balance performance and generalization. We tuned dropout values (e.g., 0.01) to prevent overfitting while preserving important temporal dependencies. Additionally, we tested varying LSTM layer sizes (e.g., 64 and 32 units) and added a dense hidden layer with 16 neurons to improve feature extraction. To avoid excessive training and minimize overfitting, we applied early exit (early stopping), which monitors validation loss and stops training if no improvement is observed for a set number of epochs. The figures show that early stopping was triggered when the validation error started increasing, preventing further overfitting.

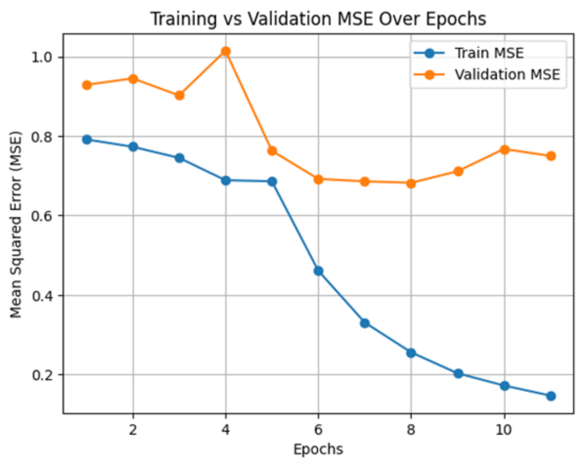


Figure 3: LSTM: Training vs Validation MSE

6.3.3 BERT

BERT is a large-scale language representation model that is pre-trained on massive corpora, enabling it to capture nuanced language patterns. The key innovation behind BERT is its bidirectional attention mechanism—during training, it considers words both to the left and right of a target token, producing richer contextual embeddings compared to unidirectional models. In our code, we utilize a pre-trained "bert-base-uncased" model, adapting it for regression by replacing the final classification layer with a single output neuron. We rely on a dedicated tokenizer to handle WordPiece tokenization, attention masks, and other inputs required by the transformer architecture. Once the data is prepared, the model is fine-tuned using MSE loss, with a learning rate typically set to 2e-5. Due to hardware constraints and the large memory demands of BERT, we apply the model to only a subset of the dataset during training; even so, the model demonstrates strong performance, significantly narrowing the gap between predicted and true ratings. That said, we found the training process to be notably time-consuming, emphasizing the trade-off between computational overhead and predictive accuracy when using advanced transformer-based approaches.

Optimization: Performance could potentially be enhanced by systematically tuning hyperparameters such as learning rate, batch size, and the number of epochs. However, due to computational limitations and hardware constraints, we did not have sufficient resources to perform extensive hyperparameter optimization experiments with BERT. Therefore, our reported BERT model performance represents an initial implementation, and future work could substantially improve results by conducting comprehensive hyperparameter tuning and optimization experiments.

6.4 Evaluation & Performance

Text Preprocessor	Model	val_mse
N/A	Mean (baseline)	0.8991
NLTK	Random Forest	0.7051
Spacy	Random Forest	0.6459
NLTK	LSTM	0.6377
Spacy	LSTM	0.6824
N/A	BERT	0.4237

Table 2: Validation MSE of different preprocessors and models

6.5 Testing on Wuxiaworld Comments Result

To evaluate the effectiveness of our model and validate our hypothesis, we conducted a random manual selection of comments from Wuxiaworld, covering positive, neutral, and negative reviews. These selected comments were then processed using our trained model to predict their ratings.

Table 3: Model prediction vs. Manual prediction

Sentences	**Model prediction score**	**Manual prediction**
"kay the image of deculein standing on a snowy rocky outcrop in a long coat whipping about in the wind is just... so exquisitely pituresque, it's so good!"	4.525612	Positive
"This just seems way better overall, specially because he started dissecting his attributes. He could use it to create the strongest mental power related magic to fight demons with, or even just implement mental damage to his usual attacks."	4.208149	Positive
"Thanks for the chapter :-)"	4.7300425	Positive
"Enjoyed reading the chapter very much."	4.6190324	Positive
"Aw.....I wanted his buds to go with him...he's so lonely!"	2.9653027	Neutral
"I'm finally all caught up, been busy ever since the start of the year..."	4.735861	Neutral
"That sly fox...of course Luo would be ok even after getting a hole through his chest lol"	2.3948398	Neutral
"I feel like this is unnecessary. He has never had a time where his physical or mental endurance was lacking. I can't picture the so-called next level."	2.231214	Negative
"Is duct tape really that legendary? I don't get it. . . ."	3.3907878	Negative
"He Yong might very well not know how to count, he is after all a martial blockhead and has taken many hits to the brain"	2.187003	Negative
"It really bugs me that he always just gives the things away, he doesn't leave them as legacies like everything HE foumd, he doesn't set theme up in a trial that is designed to temper the growth of the younger generations... Just tossed his junk to the needy and leaves."	2.0622756	Negative
"I only need to read the title to summarize the chapter.. i dont even need to read it"	4.438251	Negative
"Wow this novel is still a thing ? I just checked the most read in this site and didn't expect MGA in the top. LoL. I've read the raw but already abandoned the novel months ago because the shitty characters development in this novel. No maturity and so many stupid people just to make the MC looks great but in reality this is still stupid and unnecessary. No plot, no creativity, no consistency, many plot holes, fight like a math no skill just pure math but more mockery, useless words in a whole chapter, shiity writing skill i bet you guys can't remember the name of his friends anymore and even his girlfriends."	1.4369215	Negative

6.5.1 Observation

Observation: From the results presented in the table, it is evident that our model performs reasonably well, successfully analyzing the sentiment of most reader comments and assigning reasonable scores. The model is particularly effective in distinguishing highly positive and highly negative comments, aligning well with human perception. However, we also observed some misclassifications in specific cases:

- Some neutral comments received unexpectedly high scores, indicating that the model occasionally overestimates their sentiment.
- Certain negative comments were misclassified as neutral or even assigned higher ratings than expected.
- The model seems slightly biased toward positive predictions, leading to inflated scores in some cases.

6.5.2 Possible Causes for Model Errors and Future Improvements

One of the main reasons for the model's misclassification is the limited size of the training dataset. Due to computational constraints, we only used 20,000 samples for training. While this dataset provides a solid foundation for building a baseline model, it may not be sufficient to capture the full complexity of human sentiment in different contexts. Expanding the dataset with a larger and more diverse set of reviews could significantly enhance the model's accuracy and robustness. Another key issue stems from data imbalance in the training set. Since our dataset was sourced from Amazon book reviews, where most books have undergone professional editorial processes before publication, the majority of reviews tend to be overwhelmingly positive. Our Exploratory Data Analysis (EDA) revealed that a disproportionate number of training samples had 5-star ratings, which likely influenced the model to favor higher scores. As a result, the model sometimes assigns overly optimistic ratings to neutral or slightly negative comments. Addressing this bias by incorporating more negative and neutral samples in the training phase would help the model learn a more balanced sentiment representation. Additionally, there is a stylistic difference between Amazon reviews and Wuxiaworld comments, which may contribute to misinterpretations. Wuxiaworld comments are often shorter, more informal, and sometimes sarcastic, unlike Amazon reviews that tend to be well-structured and descriptive. This difference in writing style can confuse the model, leading to incorrect predictions. Future improvements should focus on fine-tuning the model with domain-specific data from similar platforms to better understand informal language, internet slang, and context-dependent sentiment. To further enhance the model's performance, we propose adopting more advanced deep learning techniques. Utilizing Bidirectional LSTMs (BiLSTM) or Transformer-based models such as BERT or GPT-based sentiment analysis could provide better text representations and improve the model's ability to capture nuanced sentiment variations. Additionally, implementing data augmentation strategies to artificially generate more diverse training samples could help mitigate class imbalance issues. By incorporating these improvements, we aim to refine the model's accuracy, reduce biases, and ensure that it performs well across a broader range of real-world user comments.

7 Literature Survey

The dataset used in this project, specifically Amazon reviews, has indeed been studied by multiple prior works for tasks such as sentiment classification, sentiment scoring, and rating prediction. Generally, this dataset has been employed to assess product satisfaction or to predict consumer behavior patterns based on textual sentiments expressed in user reviews. Existing literature tackling sentiment analysis of short, informal texts, such as comments and social media posts, has generally employed deep learning models including CNN, LSTM, BiLSTM, and Transformer-based architectures (e.g., BERT, GPT).[5] The state-of-the-art in sentiment analysis increasingly favors transformer-based models like BERT, as these capture context more effectively than traditional embeddings or simpler neural networks.[6] Our project uniquely applies predictive models trained on full-book reviews from Amazon to chapter-specific reader comments on serialized fiction platforms such as Wuxiaworld. Our work introduces a real-time metric of reader satisfaction, offering practical tools for authors in the rapidly evolving digital publishing landscape. The conclusions drawn from this project

[5]https://orenshapira.medium.com/using-neural-networks-and-nlp-word-embedding-to-predict-amazon-user-review-sentiment-28156f69e1e1

[6]https://huggingface.co/blog/bert-101

largely align with previous research: while simpler models (e.g., DNN with BOW, Random Forest) demonstrate reasonable performance, advanced neural architectures like BERT or Random Forest models combined with TF-IDF vectorization typically show superior predictive performance. Nevertheless, the project's results highlight the challenges of dataset imbalance and textual style differences, reflecting similar findings reported in prior work. The observed model overfitting and biases toward positive predictions echo common challenges faced in related sentiment analysis research.

8 Conclusion

In conclusion, the integration of sentiment analysis into serialized novel publishing effectively addresses the challenge of lacking chapter-specific rating mechanisms on serialized platforms. By utilizing sentiment analysis models trained on extensive full-book reviews from platforms like Amazon, it becomes feasible to accurately estimate chapter-specific reader satisfaction from textual comments alone. The study demonstrates that methods such as TF-IDF vectorization combined with machine learning algorithms—especially Random Forest and advanced deep learning models like BERT—can provide reliable predictions aligning closely with human perception. Despite some limitations in dataset size and inherent positive-rating biases, these challenges can be mitigated by expanding training data and fine-tuning models on domain-specific textual data. Ultimately, incorporating sentiment analysis empowers authors to proactively respond to reader feedback, continuously optimize their storytelling, and significantly improve reader retention and engagement in serialized fiction platforms.

9 Additional work: demo

This application is built using **Streamlit**, allowing for an intuitive and interactive web-based experience. The app takes user input, processes it through a fine-tuned BERT model, and outputs a predicted rating between 1 and 5 stars.

Key Technologies Used:

- **Streamlit** – Provides a simple and interactive user interface.
- **PyTorch** – Handles deep learning computations and inference.
- **SafeTensors** – Loads the model efficiently from a pre-saved directory.

How the App Works:

- User Input: The user enters a review in a text box.
- Model Processing: The text is tokenized and passed through the fine-tuned BERT model.
- Rating Prediction: The model predicts a numerical rating (normalized to a 1-5 scale).
- Result Display: The rating is displayed on the interface.

9.1 Referencing the Pretrained BERT Model

The app references a fine-tuned BERT model stored during previous training. Instead of retraining, it directly loads the model and tokenizer. This ensures fast inference without needing additional training. With this approach, the app efficiently predicts ratings from text-based reviews, providing a real-time and user-friendly sentiment analysis experience.

The code for demo as well as the video of presentation have been uploaded to Github .

10 Citations

Below are the references used in this report:

- Unknown. *The Impact of Serialized Books on Reader Engagement Statistics*. Available at: https://newbookrecommendation.com/the-impact-of-serialized-books-on-reader-engagement-statistics. Accessed: March 16, 2025.
- Horror Tree. *The Rise of Serialized Fiction and Its Impact on Readers*. Available at: https://horrortree.com/the-rise-of-serialized-fiction-and-its-impact-on-readers. Accessed: March 16, 2025.
- Content Square. *4 Sentiment Analysis Examples to Help You Improve CX*. Available at: https://contentsquare.com/guides/sentiment-analysis/examples. Accessed: March 16, 2025.
- Hou, Yupeng, Li, Jiacheng, He, Zhankui, Yan, An, Chen, Xiusi, and McAuley, Julian. *Bridging Language and Items for Retrieval and Recommendation*. arXiv preprint arXiv:2403.03952, 2024.

- Oren Shapira. *Using Neural Networks and NLP Word Embedding to Predict Amazon User Review Sentiment*. Available at: https://orenshapira.medium.com/using-neural-networks-and-nlp-word-embedding-to-predict-amazon-user-review-sentiment-28156f69e1e1. Published: October 23, 2022.
- Britney Muller. *BERT 101*. Hugging Face Blog. Available at: https://huggingface.co/blog/bert-101. Published: March 2, 2022.