Amazon Book Review Sentiment Regression

Dinh Dang Khoa Tran

1. Introduction

Predicting the sentiment expressed in customer reviews has become an essential task in natural language processing, particularly in domains where user feedback plays a critical role in shaping product perception and trust. Rather than simplifying sentiment into binary or categorical classes, this report approaches sentiment prediction as a regression problem—aiming to estimate a continuous score that corresponds to the star rating given by a reviewer on a 1-to-5 scale. This allows for capturing more nuanced emotional and evaluative signals embedded in natural language.

To address this task, a range of machine learning models is employed, including gradient boosting methods such as LightGBM, as well as neural network architectures like multilayer perceptrons (MLPs) and recurrent models (LSTM). The effectiveness of these models is evaluated across various embedding techniques, including TF-IDF, Word2Vec, GloVe, fastText. Each combination of embedding and model is assessed based on its ability to approximate user rating behavior from review content.

This report begins by detailing the process of dataset curation, including selection criteria, data sampling, and the application of various preprocessing techniques to prepare the text for modeling. It then presents a comparative analysis of several embedding strategies and their relevance to this sentiment regression task. Following that, each model architecture is discussed along with its respective performance and insights. Certain limitations specific to this dataset and problem space are also addressed, providing context for modeling constraints and interpretation challenges. The report concludes with a summary of key findings, evaluation of the best-performing models, and recommendations for future improvement.

The outputs of this work have practical relevance for tasks such as automated sentiment-based rating systems and other contexts where fine-grained regression-style sentiment scoring is desirable.

1.1 Context/ Dataset

This study is grounded in the context of Amazon's online review system, specifically within the Books category, where user opinions are often rich, subjective, and linguistically diverse. Unlike reviews for products such as electronics or appliances, book reviews frequently contain personal reflections, emotional reactions, and interpretive commentary, making them ideal for testing the limits of sentiment modeling.

The dataset used in this work is derived from the Amazon Customer Reviews corpus. The following fields were included:

Column Name	Description
marketplace	The two-letter country code of the marketplace where the review was written.
customer_id	An anonymous customer identifier for grouping reviews from the same user.
review_id	A unique identifier for each review.
product_id	The unique identifier of the product reviewed.
product_parent	A grouping identifier for variations of the same product.
product_title	The title of the book being reviewed.
product_category	The broad category of the product, which is “Books” in this case.
star_rating	The user-assigned rating from 1 to 5 stars (target variable for prediction).
helpful_votes	Number of users who found the review helpful.
total_votes	Total number of votes on the helpfulness of the review.
vine	Indicates if the review was part of the Vine program (paid review program).
verified_purchase	Indicates whether the product was purchased through Amazon.
review_headline	A short headline summarizing the review.
review_body	The full text of the review (main input feature).
review_date	The date when the review was submitted.

2. Method

This section outlines the end-to-end pipeline used to prepare the data and build predictive models. It includes feature selection, comprehensive text preprocessing, embedding strategies for textual representation, and the machine learning algorithms applied for rating prediction.

2.1 Feature Selection

For the purpose of sentiment regression, several fields in the original dataset were deemed unnecessary due to their limited contribution to the textual understanding of user sentiment. To enhance model reliability and minimize noise, only reviews associated with verified purchases were retained, as these are less likely to contain manipulated or fraudulent content.

Among the available features, only three columns were selected for modeling: star_rating (the target variable), review_headline, and review_body. Both review_headline and review_body consist of free-form text and often contain overlapping sentiment cues. To streamline input representation and reduce feature dimensionality, the two text fields were concatenated with a space delimiter to form a unified review text used as the primary input for embedding and modeling.

2.2 Data Preprocessing

To ensure consistency and improve model performance, several normalization and noise removal techniques were applied to the text data before embedding and training.

Text Normalization

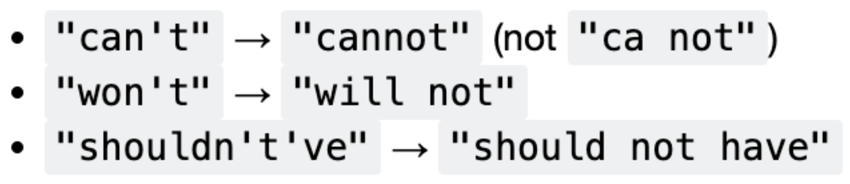
All characters in the dataset were converted to lowercase to eliminate case sensitivity, as words like “Book” and “book” should be treated identically in sentiment analysis. In addition, all punctuation and excess whitespace were replaced with single-space delimiters to remove formatting inconsistencies and reduce noise in tokenization.

HTML Tag Removal

Since many of the reviews originated from scraped web content, they contained HTML-specific tokens such as <br /> and &nbsp;. These elements do not contribute to the semantic content of the text and were removed using the BeautifulSoup HTML parser, ensuring only meaningful plain text remained.

Contraction Expansion

To handle informal language more effectively, all contractions were expanded into their full forms using a custom regular expression-based method. For instance, “don’t” was converted to “do not.” Special cases such as “can’t” (which should become “cannot” rather than “ca not”) were manually handled to preserve semantic correctness. This step helped standardize the lexical form of input words, which benefits both tokenization and embedding.



- `"can't"` → `"cannot"` (not `"ca not"`)
- `"won't"` → `"will not"`
- `"shouldn't've"` → `"should not have"`

Stop-word Removal with Negation Preservation

Common English stopwords—such as “the,” “is,” and “are”—were removed using NLTK’s predefined stopword list to reduce vocabulary size and eliminate non-informative tokens. However, certain negation words like “not” and “no” were deliberately retained, as they significantly alter sentiment polarity (e.g., “not good” vs. “good”).

Lemmatization

To reduce words to their base dictionary forms while preserving meaning, lemmatization was applied using spaCy. This process helps group related word variants (e.g., “running” → “run”) without the overly aggressive truncation introduced by stemming. Lemmatized text was used in all traditional ML pipelines, while the original, unstemmed tokens were retained for embedding methods like fastText.

2.3 Transforming Text into Word (Word Embedding)

After preprocessing the review text, various word embedding techniques were applied to transform the cleaned textual data into a numerical format suitable for machine learning models. Both classical and modern embedding approaches were tested to evaluate their effectiveness in capturing sentiment-related features. This includes traditional frequency-based methods like TF-IDF, as well as dense vector representations such as Word2Vec, GloVe, and fastText. Each embedding method was chosen based on its ability to represent contextual information, and their performance was compared in downstream regression tasks.

2.3.1 TF-IDF

TF-IDF was selected as the first embedding approach due to its simplicity, computational efficiency, and effectiveness in representing the prominence of terms within a document. In the context of book reviews—where word choice and frequency often carry strong sentiment signals—TF-IDF can highlight relevant patterns without requiring deep semantic understanding.

The TF-IDF score is computed as:

$$\mathbf{TF\text{-}IDF(t, d) = TF(t, d) \times \log(N / DF(t))}$$

Where:

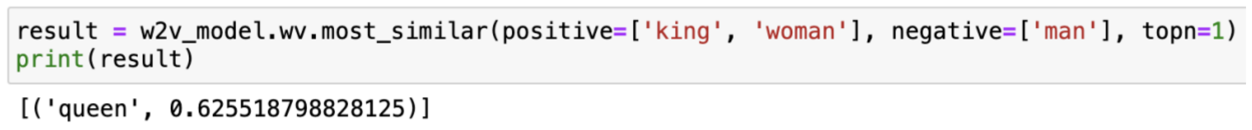
- ***TF(t, d)*** *is the number of times term t appears in document d,*
- ***DF(t)*** *is the number of documents containing the term t,*
- ***N*** *is the total number of documents.*

For this study, unigrams and bigrams were extracted from the preprocessed review text, and the top 5,000 features were retained to balance dimensionality and expressiveness. TF-IDF served as a strong baseline for comparison with more advanced embeddings, especially when used with models like LightGBM that handle sparse input well.

2.3.2 Word2Vec

To capture semantic relationships between words, a custom Word2Vec model was trained directly on the tokenized review dataset. Unlike TF-IDF, which only captures frequency-based importance, Word2Vec generates dense vector representations where words with similar meanings appear closer in vector space.

This property is especially valuable in modeling book reviews, which often contain nuanced and domain-specific vocabulary.



```
result = w2v_model.wv.most_similar(positive=['king', 'woman'], negative=['man'], topn=1)
print(result)
```

```
[('queen', 0.625518798828125)]
```

The model was trained using the skip-gram architecture with a window size of 5 and vector dimensionality of 100. Tokens were derived from lemmatized review text, and rare words with frequency below 5 were discarded to reduce noise. Each review was converted into a fixed-size feature vector by averaging the Word2Vec embeddings of all valid tokens.

2.3.3 Fasttext

FastText extends Word2Vec by representing each word as a collection of character-level n-grams, allowing it to better capture rare and misspelled words. Lemmatized tokens from each review were used as training data. The model was configured with a vector size of 100, a context window of 5, and a minimum word frequency threshold of 5. The skip-gram architecture (sg=1) was selected to prioritize semantic similarity in sparse text.

2.3.4 GloVe

GloVe embeddings were incorporated to provide dense vector representations of words based on global co-occurrence statistics. For this project, 100-dimensional pre-trained GloVe vectors (trained on Wikipedia and Gigaword) were used to encode the lemmatized review text. Each review was transformed into a fixed-size vector by averaging the embeddings of all recognized tokens.

Unlike context-free methods like TF-IDF, GloVe captures semantic relationships between words, allowing similar words to share similar vector spaces. However, as it is a static embedding model, it does not account for word meaning variation based on context or domain. In this setup, GloVe served as a benchmark for comparing traditional and domain-adaptive embedding strategies.

2.2 Applying Model

To evaluate sentiment regression performance, each embedding technique was paired with one or more machine learning models, forming a set of embedding-model combinations. The dataset was split using an 80/20 train-test ratio, and models were trained on the embedded review vectors derived from TF-IDF, Word2Vec, fastText, and GloVe. These embeddings were then used as inputs to different learning architectures, including LightGBM, multilayer perceptrons (MLPs), and recurrent neural networks (LSTM). This approach enabled a systematic comparison of how different textual representations interact with various modeling strategies in predicting continuous sentiment scores.

2.2.1 TF-IDF + LightGBM Regressor

As a strong baseline, the TF-IDF representation was paired with LightGBM, a gradient boosting framework optimized for efficiency and scalability. The sparse TF-IDF vectors, consisting of the top 5,000 unigram and bigram features, were well-suited for tree-based models like LightGBM that naturally handle high-dimensional input. The model was trained on the TF-IDF vectors derived from the concatenated review headline and body, using default hyperparameters with early stopping to avoid overfitting. This combination achieved a solid performance, providing a reference point for evaluating the effectiveness of more semantically rich embedding methods.

- **Features**: 5000 TF-IDF features (1–2 grams)
- **Train size**: ~160,000 samples
- **Starting score**: 4.17 stars
- **RMSE**: **0.9497**

2.2.2 Word2Vec + LightGBM Regressor

To leverage richer semantic information, the custom-trained Word2Vec embeddings were paired with LightGBM for regression. Each review was represented as the average of its word embeddings, using a 100-dimensional skip-gram Word2Vec model trained on the lemmatized review corpus. These dense vectors preserved contextual similarity between words, enabling the model to generalize better across semantically related expressions.

The averaged Word2Vec vectors were fed into LightGBM without dimensionality reduction, as the compact embedding size allowed for efficient training. This combination yielded the lowest RMSE among all traditional embedding approaches, demonstrating that task-specific embeddings trained on in-domain data can significantly enhance model performance in sentiment regression.

- **Features used**: 100 (vector size)
- **Train samples**: ~160,000
- **Starting score**: 4.17 stars
- **RMSE**: **0.9239**

2.2.3 GloVe + LightGBM Regressor

In this configuration, pre-trained GloVe embeddings were combined with LightGBM to evaluate how general-purpose semantic vectors perform on the regression task. Each review was encoded as the average of its 100-dimensional GloVe word vectors, computed from tokens present in the embedding vocabulary. Unlike Word2Vec, which was trained on the same domain, GloVe embeddings were fixed and not adapted to the review corpus.

Although GloVe provided a meaningful semantic foundation, its static nature and lack of domain adaptation limited its effectiveness. When passed into LightGBM, the model underperformed relative to Word2Vec, suggesting that embeddings trained on general corpora may miss domain-specific sentiment signals present in book reviews.

- **Features used: 100 (vector size)**
- **Train samples: ~160,000**
- **Starting score: 4.17 stars**
- **RMSE: 1.0369**

2.2.4 GloVe + MLP

To evaluate the performance of dense embeddings with a neural architecture, GloVe vectors were also used as input to a multilayer perceptron (MLP). As before, each review was represented by the average of its 100-dimensional GloVe embeddings. These vectors were passed to an MLP consisting of fully connected layers with ReLU activation and dropout regularization to mitigate overfitting.

This setup aimed to explore whether a non-linear model could better leverage the semantic information embedded in GloVe vectors. While the MLP demonstrated slightly better performance than the GloVe + LightGBM combination, it still fell short of the results achieved by Word2Vec and fastText models, highlighting the limitations of static, pre-trained embeddings in capturing review-specific sentiment patterns.

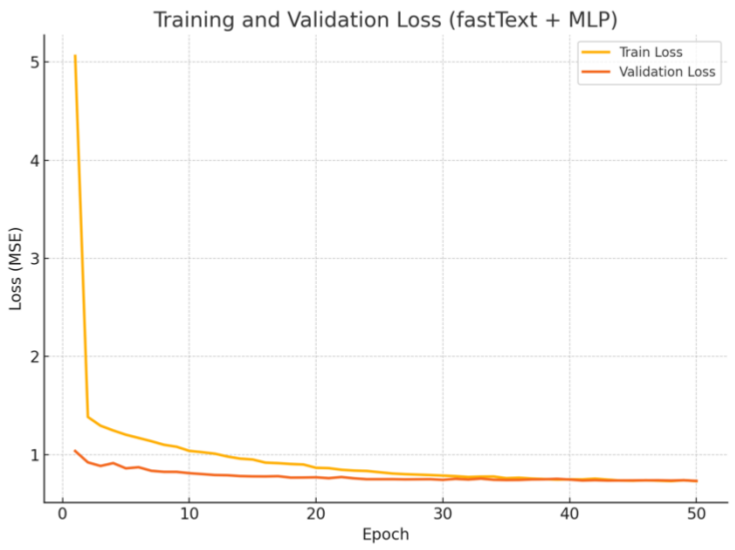
- **Features used**: 100 (vector size)
- **Train samples**: ~160,000
- **Starting score**: 4.17 stars
- **RMSE**: **1.0369**

2.2.5 fastText + MLP

In this configuration, dense vectors from the custom-trained fastText model were used as input to a multilayer perceptron (MLP). Each review was represented by the average of its fastText word embeddings, which capture subword information and are more robust to misspellings and rare words. The MLP architecture consisted of multiple fully connected layers with ReLU activation and dropout, allowing the model to learn non-linear patterns in the sentiment data.

This combination aimed to evaluate whether fastText's subword-aware embeddings could enhance performance when paired with a deep feedforward model. While results were competitive, the overall RMSE remained slightly higher than that of Word2Vec + LightGBM, suggesting that sequential models like LSTM may be better suited for capturing the structure of fastText embeddings.

- **Train samples**: ~160,000
- RMSE: **0.9733**



2.2.6 GloVe + Simple RNN

To incorporate temporal dynamics of text, GloVe embeddings were used as input to a simple recurrent neural network (RNN). Each review was tokenized and mapped to its corresponding 100-dimensional GloVe vectors, forming a sequence of word embeddings. These sequences were fed into an RNN layer to capture word order and sequential patterns that static averaging methods ignore.

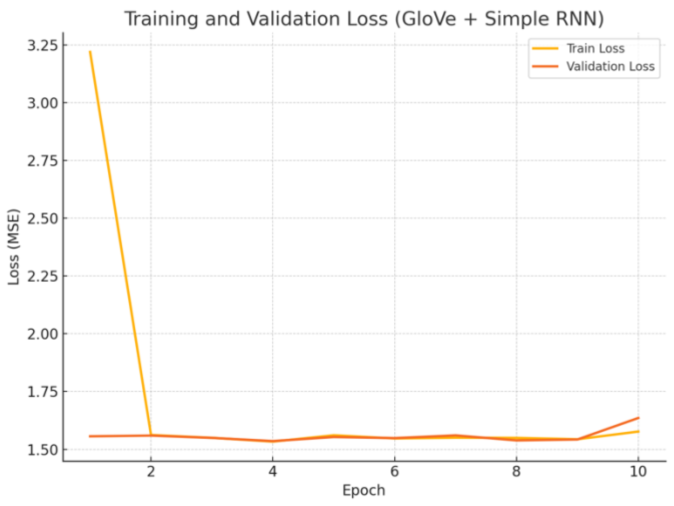
While the RNN architecture allowed the model to account for local context and word dependencies, its performance was constrained by the limitations of GloVe as a static embedding method. Without domain-specific adaptation, the model struggled to consistently align embedding semantics with sentiment expression in the book reviews, resulting in moderate improvement over non-sequential baselines.

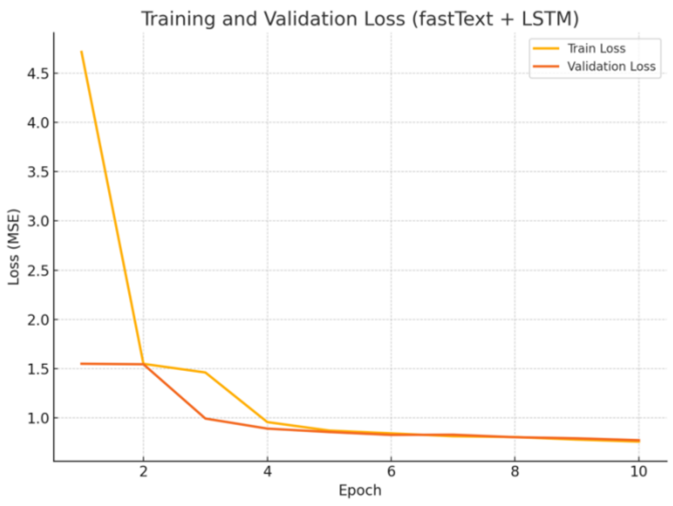
- **GloVe + SimpleRNN RMSE: 1.2469**

2.2.7 GloVe + LSTM

To better capture sequential dependencies in review text, GloVe embeddings were combined with a Long Short-Term Memory (LSTM) network. Pre-trained 100-dimensional GloVe vectors were used to initialize word embeddings, and each review was tokenized into sequences and padded to a uniform length. These sequences were then passed through an LSTM layer, allowing the model to learn temporal patterns and long-range dependencies that static averaging fails to capture.

The use of LSTM was intended to explore how well a recurrent architecture could leverage the semantic structure of reviews beyond simple word frequency. While the GloVe vectors provided solid lexical grounding, their static nature and limited alignment with the book review domain may have constrained the LSTM's ability to model more nuanced sentiment transitions. Nevertheless, this combination outperformed simpler GloVe-based models and showed that LSTM can enhance contextual understanding when paired with dense, pretrained embeddings.





3. Result and Discussion

The final RMSE scores across all embedding-model combinations are summarized in Figure X. Root Mean Squared Error (RMSE) was used as the evaluation metric to measure the average difference between predicted and actual star ratings.

Embedding	**Model**	**RMSE**
TF-IDF	LightGBM	0.9497
Word2Vec (Averaged)	LightGBM	0.9239
GloVe (Averaged)	LightGBM	1.0369
GloVe (Averaged)	MLP	0.9733
fastText (Averaged)	MLP	**0.8501** ✅ Best

Embedding	Model	RMSE
GloVe	SimpleRNN	1.2270
GloVe	LSTM	0.8604

The best performance was achieved by **fastText + MLP** (RMSE = 0.8501), demonstrating the advantage of subword-level embeddings in handling diverse and informal review text. **GloVe + LSTM** followed closely, showing that sequential models benefit from richer word representations even if pre-trained.

Word2Vec + LightGBM also delivered strong results, validating the effectiveness of domain-specific embeddings with traditional models. In contrast, **GloVe + SimpleRNN** performed the worst, likely due to limited modeling capacity and the static nature of GloVe vectors.

Overall, the results confirm that pairing subword-aware embeddings or sequence models with meaningful representations significantly improves sentiment regression compared to traditional baselines like TF-IDF.

4. Conclusion

This project successfully built a regression-based sentiment prediction system capable of estimating Amazon book review ratings from raw text. By experimenting with various combinations of word embeddings and machine learning models, I was able to identify the most effective configuration — fastText embeddings paired with an MLP — which achieved the best RMSE score. Additionally, a final demonstration showed that the models could generalize well to new inputs and return interpretable sentiment scores.

Through this process, I gained practical experience with key NLP components such as text preprocessing, tokenization, lemmatization, and embedding. I also deepened my understanding of how different embedding strategies (e.g., static vs. subword vs. domain-trained) affect model performance. Working with both traditional models (like LightGBM) and deep learning architectures (such as LSTM and MLP) taught me how to align data representation with model structure. Overall, the project provided a comprehensive, hands-on learning experience in building end-to-end machine learning systems for real-world language data.

Reference:

Stanford CS224n Note 1: Introduction and Word2Vec:
https://web.stanford.edu/class/cs224n/readings/cs224n_winter2023_lecture1_notes_draft.pdf

Stanford CS224n Note 2: Word Vectors II: GloVe, Evaluation and Training
https://web.stanford.edu/class/cs224n/readings/cs224n-2019-notes02-wordvecs2.pdf

Product review prediction of Amazon products using machine learning models by Viktoria Vida:
https://arno.uvt.nl/show.cgi?fid=161823