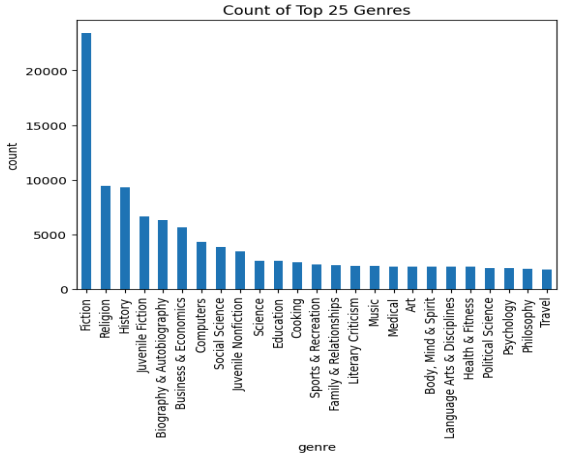
GROUP 11

Introduction

In today's digital shopping landscape, platforms like Amazon offer vast book selections, overwhelming readers seeking personalized recommendations. Our project aims to address this challenge by leveraging the Amazon Books Reviews dataset to develop a recommendation system. By analyzing user feedback and book attributes, we seek to provide tailored book suggestions, enhancing readers' literary journey. "Amazon Books Reviews" combines Amazon Review and Google Books API data, comprising millions of user reviews on over 200,000 books. It includes user-specific details and book attributes, forming a rich resource for analysis.

Exploratory Data Analysis

These are some of the important insights that we derived from the exploratory data analysis.



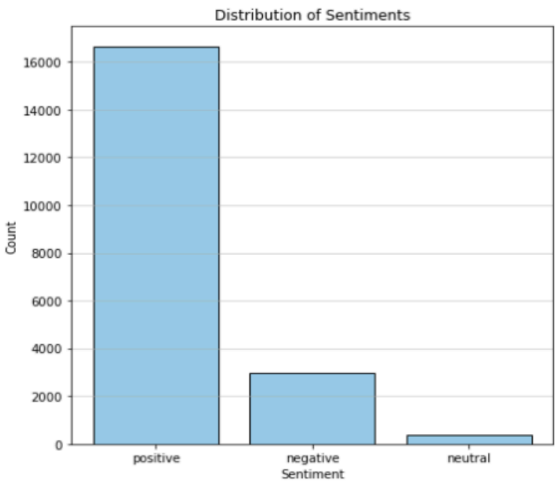


Figure 1: Fiction has the greatest count of masterpieces
Figure 2: Most of the texts are positive and only a few of the texts are neutral or negative.

NLP - Sentiment Analysis

As part of the project, we have performed Sentiment Analysis using the columns 'review/text' (the textual data corresponding to reviews), and 'Sentiment' (label indicating sentiment of review as 'Negative', 'Neutral', and Positive). The various techniques employed are as follows:

1. **Long Short Term Memory (LSTM) Neural Network -** Due to the enormous size of the dataset, 3 LSTM layers have been added to the LSTM network, along with Dropout layers, and Batch Normalization layers to prevent overfitting. The model achieves an accuracy of 84.39% on the Test Set, which is decent but not as good as the Logistic Regression Model. The model achieves high precision, recall and f1-scores for the majority class, but poor precision, recall and f1-scores for the minority classes ('Neutral' and 'Negative'), simply because the dataset is very huge and imbalanced. The LSTM model achieves a high ROC AUC, but a low PR AUC for the minority classes. Since the LSTM model is trained on a huge and imbalanced dataset, the rarity of the minority class leads to less training data, therefore the LSTM model learns to recognize instances of the minority class but with less confidence in precision.

2. **Logistic Regression -** We performed a Grid Search to find the best set of hyperparameters for the Logistic Regression model. The model achieves an accuracy of 87%, which is good compared to the rest of the models. Upon evaluating the best estimator from the Grid Search, we see that the precision, recall, and f1-scores for the majority class ('Positive') were high. But it achieves poor precision, recall and f1-scores for the 'Neutral' and 'Negative' classes. This variation is simply because the dataset is very huge and imbalanced. Moreover, we see that the ROC-AUC of the minority class ('Neutral') is the highest among the three classes. This clearly implies that the Logistic Regression Model is able to distinguish instances of the Minority Classes. Thus, the Logistic Regression Model has learned to generalize to the patterns present in the data) and has not over-fitted.
3. **Naive Bayes -** We trained a Multinomial Naive Bayes model. The model accuracy is 66%, which is poor compared to other models we tested. The model achieved a 94% precision score for the positive class, while precision for negative and neutral classes are 37% and 7% respectively. This shows that this model is unable to generalize over the imbalanced dataset.
4. **Random Forest** - We trained a random forest model with 100 estimators and a maximum depth of 30 to mitigate the impact of overfitting. The random forest model had a good accuracy of 85%. While precision is high across the 3 classes, recall scores for negative and neutral classes are poor. Couple with a near-perfect recall score for the positive class, this shows the model was not generalizing well over this imbalanced dataset.

Scores for different sentiment analysis models tested

Model	Accuracy	Weighted Average Precision	Weighted Average Recall	Weighted Average F1	Weighted Average ROC-AUC
Naive Bayes	0.66	0.84	0.66	0.72	0.81
Random Forest	0.85	**0.87**	0.85	0.80	**0.89**
Logistic Regression	**0.87**	0.85	**0.87**	**0.85**	**0.89**
LSTM	0.84	0.82	0.84	0.79	0.80

Recommendation System

1) Collaborative Filtering Recommendation System

The collaborative filtering model is used to predict user preferences and recommend items based on the user's past interaction data, and two collaborative filtering modes are implemented in this project. The first model is a basic approach to collaborative filtering: Based on user similarities, the model efficiently provides recommending book titles. The second model is implemented by using PyTorch to build neural networks, in large-scale environments. The integration of embedding layers and similarity calculations effectively predicts ratings and provides recommending book titles.

a) **Model 1**

Architecture: The dataset encompasses user reviews and ratings of a wide range of titles. From these reviews, key keywords are extracted to define user similarity. Subsequently, dimensionality reduction techniques are applied to the matrix of users and keywords, effectively compressing the user-item interaction space. This process is conducted with precision to retain the core aspects of user preferences. Within this reduced space, the nearest neighbors are determined through cosine similarity. Ultimately, only those titles with a review score exceeding 3.5 are recommended, ensuring a focus on quality in the suggestions provided to the users.

b) **Model 2**

Architecture: The recommendation system operates on a dataset of user-item interactions by focusing on user reviews and ratings of various titles. Separate embeddings are created for users and items, allowing the model to learn dense representations of each entity based on interaction data. The embeddings are concatenated and passed through a fully connected hidden layer with nonlinear activation (ReLU), facilitating the capture of complex interactions between user and item features. A linear output layer predicts the rating a user would give to an item, as demonstrated in the forward pass of the model. Additionally, the model supports querying similar items based on their learned embeddings. By computing cosine similarity between item embeddings, the system can suggest items that are contextually similar to a given item, enhancing user discovery and engagement.

Result: The model is trained using a batch processing approach with the PyTorch DataLoader, optimizing the prediction error between the estimated and actual ratings. The model predicts the `rating/score` of the book based on the review titles with a Mean Squared Error of 1.11 and Root Mean Squared Error of 1.0535, which is not very promising but still meaningful.

2) **Content-based Filtering Recommendation System**

For the purpose of providing recommendations for books, the book attributes that we have considered are: ‘Title’, ‘authors’ and ‘categories’. We applied TF-IDF Vectorization on the concatenation of these three columns to produce the TF-IDF matrix, whose dimensionality is reduced to 100 components using Singular Value Decomposition (SVD). 1% of the books are sampled from the dataset randomly to avoid Memory Limit Exceeded errors during the generation of cosine similarity matrix due to the enormous size of the dataset. The matrix obtained after sampling of the SVD matrix is used for the generation of the cosine similarity matrix. The cosine similarity matrix quantifies the similarity between all pairs of books in the dataset. The key idea is that books that are similar to each other, based on the book attributes ('Title', 'authors', and 'categories') have higher cosine similarity scores. Therefore, The cosine similarity matrix enables us to recommend books that are content-wise similar to a particular book.

Conclusion / Limitations

According to the wordcloud from the EDA analysis, there is no big difference between word frequency for review scores less or equal to 3 and greater than 3. The Collaborative Filtering (Model 1) and Content-Based Filtering efficiently provide recommending book titles based on the user similarity and features similarity. The Collaborative Filtering (Model 2) effectively predicts ratings with rmse of 1.0535 and provides recommending book titles. In the Collaborative Filtering Model 2, even though multiple epochs could be implemented to minimize loss and improve recommendation accuracy, but due to time and memory constraints, 1 epoch has been implemented in this project. For the NLP sentiment analysis task, we found that the logistic regression model performed the best, with the highest accuracy, F1 and AUC-ROC.

References

[1] Fang, X., Zhan, J. Sentiment analysis using product review data. *Journal of Big Data* 2, 5 (2015). https://doi.org/10.1186/s40537-015-0015-2

[2] Miller, Chris. "Build a Recommendation Engine With Collaborative Filtering." Real Python, 20 Aug. 2020, realpython.com/build-recommendation-engine-collaborative-filtering/.

[3] Bakhet, Mohamed. "Amazon Books Reviews." Kaggle, 2022, https://www.kaggle.com/datasets/mohamedbakhet/amazon-books-reviews/data.

[4] "Understanding Sentiment Analysis and Its Application." Analytics Vidhya, 14 June 2021, www.analyticsvidhya.com/blog/2021/06/nlp-sentiment-analysis/

[5] Goyal, Ansh K. "Book Recommendation System." Kaggle, 2021, https://www.kaggle.com/code/anshkgoyal/book-recommendation-system.

[6] Roy, D., Dutta, M. A systematic review and research perspective on recommender systems. *J Big Data* 9, 59 (2022). https://doi.org/10.1186/s40537-022-00592-5