Bilkent University
CS464 Machine Learning
Progress Report
Group 15

Book Recommendation System using Collaborative Filtering Neural Network with Embeddings

Group Members:

- **Ferhat Korkmaz 21901940**
- **Ömer Oktay Gültekin 21901413**
- **Utku Kurtulmuş 21903025**
- **Dilay Yigit 21602059**
- **Muhammet Oğuzhan Gültekin 21801616**

Table Of Contents

1. Introduction

The outline of our project ‘Booking Recommendation System’ is the design and development of a tailored book recommendation system that aims to refine the way readers discover new books and cultivate their literary interests. Drawing from a rich dataset sourced from Kaggle, which encapsulates over 3 million book reviews for 212,404 unique titles, our project makes use of a comprehensive Amazon review dataset, extending from May 1996 to July 2014.

Our data foundation consists of two main components: the primary file focuses on the evaluative aspect of reader feedback, while the 'Books Details' file enhances this with comprehensive information about each book obtained from the Google Books API. This combination of quantitative ratings and qualitative descriptions forms the basis for our model’s recommendation logic. During data preparation, we have selected a subset of dedicated readers—those who have reviewed a minimum of 20 books— and used their reviews to train our prediction models. We have used explicit feedback mechanism while implementing our model.

In the current phase, we have implemented collaborative filtering through a neural network with intricate neuron structures, incorporating user and item embeddings. This advanced technique allows our model to discern complex user-book interactions, forming the basis for generating personalized recommendations. The neuron structure of our collaborative filtering model enables it to capture and understand nuanced patterns in user preferences. The user and item embeddings contribute to this understanding by representing users and books in a high-dimensional space.

As we progress, our next steps include the implementation of content-based filtering, where we will leverage techniques such as cosine similarity to enhance the accuracy of our recommendations. Additionally, we plan to explore regression methods, including Linear Regression, Decision Tree Regression, and Random Forest Regression, to further refine our predictive modeling capabilities. Also, we will do hyper-tuning on parameters as we progress. These strategic next steps aim to optimize our model's performance, ensuring it evolves to meet the dynamic needs of users and delivers precise, tailored book recommendations.

The ultimate objective of this project extends beyond creating a system that enhances users' literary journeys; it also aims to contribute to the broader discussion on how machine learning can personalize digital experiences by not simply recommending similar books but also learn from other similar readers' preferences. This report will document our approach, covering aspects from data acquisition and preprocessing to model selection, coding environment, and details of the trained model in a recommendation system that is both robust and attuned to users' evolving literary preferences.

2. Data Preprocessing

The Amazon Book Review dataset contains information about 3 million book reviews for 212404 unique books. Also, the details of those books are given in the below figures [1].

	Id	Title	Price	User_id	profileName	review/helpfulness	review/score	review/time	review/summary	review/text
2971446	B000G167FA	59970	NaN	9731	Ellen C. Falkenberry "ellenf"	5/5	5.0	-1	Have you ever watched the fairies when the rai...	Although the new cover looks more like a Book ...
2006209	B00005O4HA	52409	NaN	7094	Stan Vernooy	4/5	5.0	840240000	The best mystery novel I have ever read	I have been a mystery reader for decades, and ...
2111966	B000PRURRE	30319	NaN	6939	Michael J. Tresca "Talien"	2/2	5.0	850694400	A golden piece of sci-fi pulp that shines even...	This is one of those books that seems like it'...
1359864	0671536109	62006	NaN	6939	Michael J. Tresca "Talien"	18/19	5.0	850694400	The final world on the chronology of the Star ...	This book is a must have for any serious Star ...
100284	B0001BJEG4	30320	NaN	6939	Michael J. Tresca "Talien"	2/2	5.0	850694400	A golden piece of sci-fi pulp that shines even...	This is one of those books that seems like it'...

Figure 1: Books_Rating.csv file data structure

	Title	description	authors	image	previewLink	publisher	publishedDate	infoLink	categories	ratingsCount
0	Its Only Art If Its Well Hung!	NaN	['Julie Strain']	http://books.google.com/books/content?id=DykPA...	http://books.google.nl/books?id=DykPAAAACAAJ&d...	NaN	1996	http://books.google.nl/books?id=DykPAAAACAAJ&d...	['Comics & Graphic Novels']	NaN
1	Dr. Seuss: American Icon	Philip Nel takes a fascinating look into the k...	['Philip Nel']	http://books.google.com/books/content?id=IjvHQ...	http://books.google.nl/books?id=IjvHQsCn_pgC&p...	A&C Black	2005-01-01	http://books.google.nl/books?id=IjvHQsCn_pgC&d...	['Biography & Autobiography']	NaN
2	Wonderful Worship in Smaller Churches	This resource includes twelve principles in un...	['David R. Ray']	http://books.google.com/books/content?id=2tsDA...	http://books.google.nl/books?id=2tsDAAAACAAJ&d...	NaN	2000	http://books.google.nl/books?id=2tsDAAAACAAJ&d...	['Religion']	NaN
3	Whispers of the Wicked Saints	Julia Thomas finds her life spinning out of co...	['Veronica Haddon']	http://books.google.com/books/content?id=aRSIg...	http://books.google.nl/books?id=aRSIgJIq6JwC&d...	iUniverse	2005-02	http://books.google.nl/books?id=aRSIgJIq6JwC&d...	['Fiction']	NaN
4	Nation Dance: Religion, Identity and Cultural ...	NaN	['Edward Long']		NaN	http://books.google.nl/books?id=399SPgAACAAJ&d...	NaN	2003-03-01	http://books.google.nl/books?id=399SPgAACAAJ&d...	NaN	NaN

Figure 2: Books_Data.csv file data structure

The mapping of these two different csv files was accomplished with the field “Title” as seen in Figure 3.

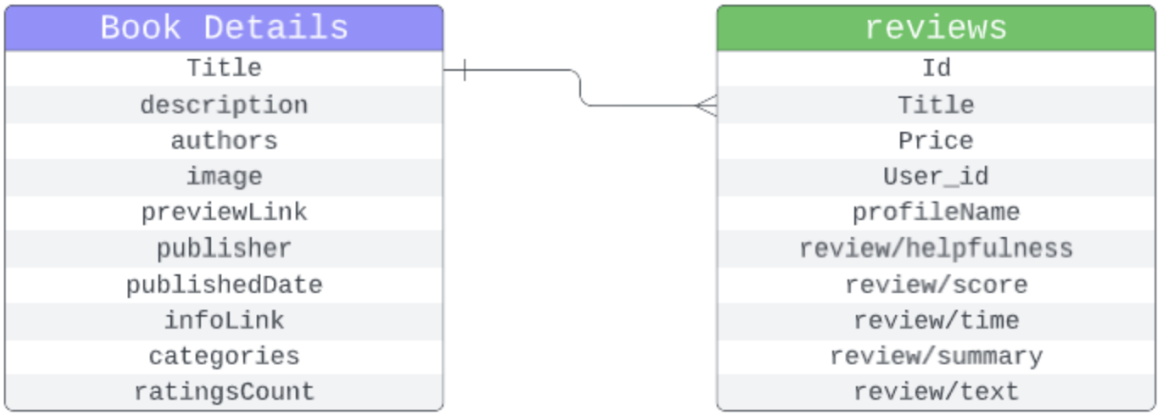

Figure 3: The diagram of files with their columns and relationship

However, for the initial model, we wanted to use Collaborative Filtering Approach and did not need most of the fields [2]. In other words, we have eliminated all of the fields other than “Title, Id, User_id, and review/score”. Therefore, we decided to adopt the explicit feedback mechanism for our model.

Also, In order to train our model with more data for each user. We wanted to eliminate some reviews. Our criteria was, if we were to use a user’s review data, that user should have at least 20 unique book reviews. As a result there are 7209 users that have reviewed at least 20 books. And 548134 unique review data. At first, we were skeptical about whether that number of reviews were enough or not. However, as we tested our model with some criteria, which will be explained later on, we got an acceptable rate of loss value and accuracy.
We have label encoded the user_id and title columns, using the label encoder of scikitlearn.

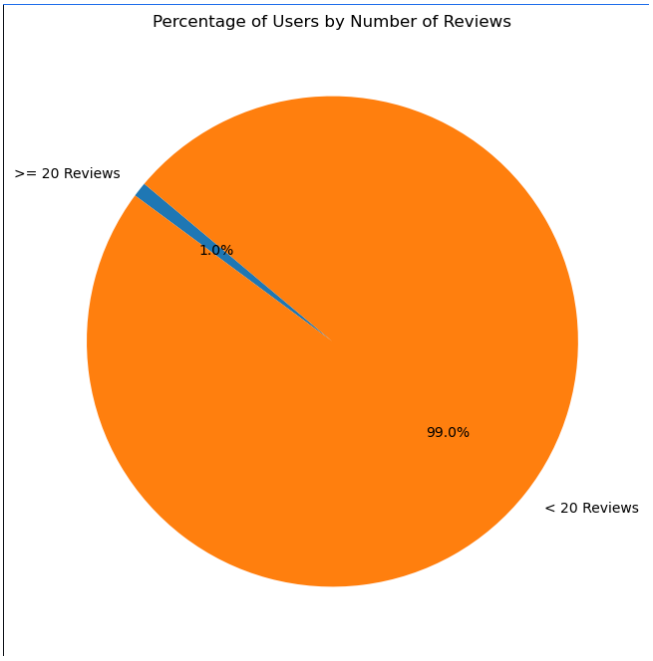

Figure 4: Pie Chart of the distribution of users
with respect to review numbers

After eliminating some of the features and the data, the last thing we had to do was splitting the dataset for training, validation, and testing purposes. We have sorted our array according to the review date, and the oldest 80 percent are reserved for the train set and the newest 10 percent for the test set. We are using an 80-10-10 split.

2.1 Training Dataset

Comprises 80% of the data, used for training the model on user ratings for books.

2.2 Validation Dataset

Consists of 10% of the data, employed to fine-tune and validate the model during training.

2.3 Test Dataset

The remaining 10%, used to assess the model's performance on new, unseen data

3. Coding Environment

In the pursuit of developing a book recommendation system, our project was scaffolded within the versatile and robust coding environment provided by Google Colab. The selection of Google Colab as our primary development platform is because it promotes collaborative work in real-time without necessitating a version control system, and it furnishes enhanced computational efficiency [3]. This environment is especially conducive for data-intensive tasks, such as training sophisticated machine learning models.

Python, acclaimed for its simplicity and power, served as the cornerstone programming language for this project [5]. Its widespread adoption and the richness of its libraries make it an ideal candidate for machine learning tasks. Within this Pythonic ecosystem, we utilized a suite of libraries tailored to our project needs:

- TensorFlow: This open-source library is renowned for its flexible ecosystem and extensive resource pool that supports machine learning and deep learning model development and deployment. TensorFlow provided the infrastructure necessary for building and training our neural network-based recommendation system [8].

- Pandas & NumPy: These libraries were instrumental for data manipulation and numerical computation, respectively. Pandas, with its high-level data structures, made data cleansing, exploration, and preprocessing both intuitive and efficient. NumPy complemented these operations by offering powerful array-processing capabilities, thereby streamlining the computational aspect of our data handling processes [6] [7].

- Python: Beyond serving as the language of choice, Python's role extended to integrating these libraries into a cohesive workflow. Its syntax and language features enabled seamless transitions between different stages of data processing and model development.

- Scikit-learn: Scikit-learn is a popular library known for its machine-learning models. However, we are using it to evaluate the performance of our model by using its accuracy metric from its metric library [11].

The amalgamation of Google Colab, TensorFlow, Pandas, NumPy, Scikit-learn, and Python established a formidable coding ecosystem, equipping us with the necessary tools to tackle the complexities of our project. This environment not only supported the technical demands of our recommendation system but also fostered a dynamic and iterative development process. The outcome is a testament to the efficacy of these tools when applied to the challenges of predictive modeling and machine learning.

4. Details of the Trained Model

4.1 Feature Embeddings

As an initial step to implementing our model, we had to describe our input data in a way that our neural network would be able to understand and process our data to find the patterns in the input data [9]. To do that, we have encoded our title and user_id data to converge them into numerical data. Then, we could have fed our model with this data. However, for our model to work more efficiently, we have needed to embed this numerical data into dense vectors with continuous numbers [9] [10]. Thus, we have an initial embedding layer where we give the encoded user and item input to the layer to get the embedded results[9]. As an initial embedding size, we have selected 20, but we might change the embedding size during the parameter tuning process. In general, increasing the embedding size can help to capture the features of the item more accurately, but it can increase the model complexity.

```
# Get the number of unique users, items, and categories
num_users = filtered_ratings_dataset['User_id'].nunique()
num_items = filtered_ratings_dataset['Title'].nunique()
```

```
# Define the embedding size
embedding_size = 20

# Define input layers for user, item, and content features
user_input = Input(shape=(1,), name='user_input')
item_input = Input(shape=(1,), name='item_input')

# Embedding layers
user_embedding = Embedding(input_dim=num_users,
output_dim=embedding_size, input_length=1)(user_input)
item_embedding = Embedding(input_dim=num_items,
output_dim=embedding_size, input_length=1)(item_input)
```

4.2 Neural Network Implementation

After the embedding layer, we have a dense layer where we give embedded inputs to our neural network. However, before giving the direct embedded result, we flatten our embedded data and concatenate them.

```
# Flatten embeddings
user_flat = Flatten()(user_embedding)
item_flat = Flatten()(item_embedding)

# Concatenate the flattened embeddings
concatenated_features = Concatenate()([user_flat, item_flat])
```

Then, we define our dense layer, with 128 neurons and a ReLu activation function. Initially, we have selected to go with one dense layer, but in the future, we can increase the layer size to increase the accuracy of our model with a cost of model complexity.

```
# Dense layer for further processing
dense_layer = Dense(128, activation='relu')(concatenated_features)
```

In the last layer of our network, we have the output layer where we get the review prediction for the given input. This layer has only one neuron that will give the output.

```
# Output layer
output_layer = Dense(1)(dense_layer)
```

As a final step, we finish our model by selecting the Adam optimizer as the model optimizer and the mean squared error (MSE) function as the loss function. In the final, we might again change them while we are doing parameter tuning and model optimization.

```
# Create the model
model = Model(inputs=[user_input, item_input], outputs=output_layer)
model.compile(optimizer='adam', loss='mean_squared_error')
```

4.3 Training Our Model

We are training our model with 10 epochs, which means we are passing the whole train dataset 10 times to improve our model at each time with backpropagation. Then, at the end of each evaluation, we see the performance improvement on both the training dataset and the validation dataset. The performance metrics are given in terms of our loss function MSE.

```
# Define the number of epochs
num_epochs = 10

model.fit(
    [
        train_set['User_id'],
        train_set['Title'],
    ],
    train_set['review/score'],
    epochs=num_epochs,
    validation_data=(
        [
            validation_set['User_id'],
            validation_set['Title'],
        ],
        validation_set['review/score']
    )
)
Epoch 1/10
13704/13704 [==============================] - 77s 5ms/step - loss:
0.8505 - val_loss: 1.0004
Epoch 2/10
13704/13704 [==============================] - 68s 5ms/step - loss:
0.5786 - val_loss: 1.0372
Epoch 3/10
13704/13704 [==============================] - 66s 5ms/step - loss:
0.4836 - val_loss: 1.1084
Epoch 4/10
```

```
13704/13704 [==============================] - 66s 5ms/step - loss:
0.4103 - val_loss: 1.1253
Epoch 5/10
13704/13704 [==============================] - 68s 5ms/step - loss:
0.3490 - val_loss: 1.1803
Epoch 6/10
13704/13704 [==============================] - 66s 5ms/step - loss:
0.2973 - val_loss: 1.2124
Epoch 7/10
13704/13704 [==============================] - 66s 5ms/step - loss:
0.2562 - val_loss: 1.2333
Epoch 8/10
13704/13704 [==============================] - 66s 5ms/step - loss:
0.2221 - val_loss: 1.2381
Epoch 9/10
13704/13704 [==============================] - 69s 5ms/step - loss:
0.1948 - val_loss: 1.2712
Epoch 10/10
13704/13704 [==============================] - 65s 5ms/step - loss:
0.1721 - val_loss: 1.2873
```

We can see that our training loss still didn't converge and we could have gone with more epochs at the final stage. However, our validation loss seems to increase as we train our data more on the training dataset, which can be a signal of overfitting.

4.4 Performance Evaluation

As a final step, we are using our test set to see the true success of our model. We are predicting the review score of each user and book combinations in the test set. We also calculate the loss function result as an initial performance evaluation metric.

```
# Evaluate the model on the test set
test_loss = model.evaluate(
    [
        test_set['User_id'],
        test_set['Title'],
    ],
    test_set['review/score']
)
print(f'Test Loss: {test_loss}')

# Make predictions on the test set
test_predictions = model.predict(
```

```
    [
        test_set['User_id'],
        test_set['Title'],
    ]
)

1713/1713 [==============================] - 4s 2ms/step - loss: 1.2794
Test Loss: 1.2794238328933716
```

We get a similar result to our validation set results as can be expected. However, to further evaluate the performance of our model we wanted to find the accuracy of our results. Calculating directly the accuracy in a regression model is not a good way to test the model. Therefore, we need to transform our regression problem into a classification problem, by defining a book review by liked or not liked label. To do that we used a binary label, where we labeled each review with a score greater than 3/5 as liked or 1, and 0 if it is less than 3/5. By doing so we get an 88.5 percent accuracy metric.

```
from sklearn.metrics import accuracy_score

# Define a threshold for classification
threshold = 3

# Convert the continuous predictions to binary (1 if predicted score >=
threshold, 0 otherwise)
binary_predictions = (test_predictions >= threshold).astype(int)

# Evaluate accuracy based on the threshold
accuracy_binary = accuracy_score((test_set['review/score'] >=
threshold).astype(int), binary_predictions)
print(f'Accuracy (Threshold={threshold}): {accuracy_binary}')

Accuracy (Threshold=3): 0.8845915277118984
```

4.4 Book Recommendation

To recommend a book to the user, we use a similar approach that we did in our test set evaluation. We recommend books by predicting their review scores and sorting them according to their score. Then we give the highest predicted books as recommendations. However, we do filter books that give a prediction lower than the liked threshold. Currently, we are doing a book recommendation by selecting 10 random books from the dataset, but we can do this for all the books in our dataset.

```
# Select a randomly encoded user ID
user_id_encoded = np.random.choice(train_set['User_id'].unique())
```

```
# Choose 10 random book titles from the dataset and encode them
random_book_titles = np.random.choice(train_set['Title'].unique(),
size=10, replace=False)

# Convert user ID and book titles to the appropriate format
user_ids_formatted = np.array([user_id_encoded] * 10, dtype='int32')
book_titles_formatted = np.array(random_book_titles, dtype='int32')

# Make predictions with the model
predicted_scores = model.predict([user_ids_formatted,
book_titles_formatted])

# Convert predicted scores and book titles to a DataFrame
recommendations_df = pd.DataFrame({
    'Book_Title': title_encoder.inverse_transform(random_book_titles),
    'Predicted_Score': predicted_scores.flatten()
})

# Filter books with a score higher than 3.0 and sort in descending
order
recommended_books =
recommendations_df[recommendations_df['Predicted_Score'] >
3.0].sort_values(by='Predicted_Score', ascending=False)

# Present recommended books and predicted scores to the user
print(f"Recommended Books for User ID (encoded): {user_id_encoded}")
print(recommended_books)
```

```
1/1 [==============================] - 0s 69ms/step
Recommended Books for User ID (encoded): 6613
                                          Book_Title  Predicted_Score
0                                   The Lifted Veil         5.014599
6  The Tao of Yiquan: The Method of Awareness in ...         4.394507
3                                         Stone Soup         4.351860
4                 North to Powder River: The Gringo          4.186153
7           Pattons Panthers the 761ST Tank Battalio         4.126636
9                                             Hunted         3.684649
1  A Counting Book with Billy & Abigail (Billy an...         3.121403
```

5. Next Steps

5.1 Integration of Content-Based Filtering

In the upcoming phases of our project, we aim to implement a content-based filtering approach to complement our collaborative filtering model. This entails leveraging book features such as descriptions, authors, categories, and publisher names to enhance the accuracy of our recommendations [2]. We plan to utilize techniques like cosine similarity and TF-IDFs to quantify the similarity between books based on their content [12]. We may add the content-based filtering either as feature embeddings directly into the neural network or as an independent filtering that selects the predetermined number of books based on content and gives it to collaborative based filtering rather than selecting 10 random books in the current implementation.

5.2 Exploring Dimensionality Reduction with PCA

To streamline the handling of high-dimensional data and potentially enhance model efficiency, we will explore the application of Principal Component Analysis (PCA). If deemed necessary, PCA will be employed to reduce the dimensions of our features while retaining essential information.

5.3 Prediction Phase Improvements

In the predictive modeling phase, we plan to incorporate regression techniques, such as Linear Regression, Decision Tree Regression, and Random Forest Regression if possible. These models will be trained to predict user ratings based on relevant features, contributing to a more comprehensive and accurate recommendation system. Otherwise, we will compare their accuracy with the current implementation with neural networks. We will also try to visualize our embeddings with the t-SNE method to get better insights [13].

5.4 Hyperparameter Tuning and Model Evaluation

As part of our iterative development process, we will engage in hyperparameter tuning to optimize the performance of our models. Parameters such as negative/positive rate, epoch count, number of neurons, number of layers and embedding dimension will be systematically adjusted and tested on validation sets. This process aims to fine-tune our models and mitigate overfitting concerns.

5.5 Iterative Development and Continuous Improvement

Our approach is iterative, allowing for continuous development and improvement. As we integrate content-based features, regression models, and conduct hyperparameter tuning,

we will closely monitor the impact on recommendation accuracy. This iterative process ensures that our recommendation system evolves to meet the dynamic needs of users, delivering precise and tailored book recommendations.

6. Workload Distribution

Most of the work is done together.

Ferhat Korkmaz: Helped implementation of data preprocessing of the code | Helped implementation of the book recommendation part of the code || Progress Report.

Ömer Oktay Gültekin: Helped implementation of the embedded model | Progress report.

Utku Kurtulmuş: Helped implementation of the embedded model | Implementation of the evaluation part of the code | Progress Report.

Dilay Yigit: Helped implementation of data preprocessing of the code | Helped implementation of the book recommendation part of the code || Progress Report.

Muhammet Oğuzhan Gültekin: Helped implementation of the embedded model | Progress report

7. References

[1] “Amazon Books Reviews”, Kaggle, 2023. [Online]. Available: https://www.kaggle.com/datasets/mohamedbakhet/amazon-books-reviews/data.

[2] “How to Build a Recommendation System”, Analytics Vidhya, 2022. [Online]. Available:https://www.analyticsvidhya.com/blog/2021/06/build-book-recommendation-system-unsupervised-learning-project/.

[3] “Collaborative Filtering Advantages & disadvantages | machine learning | google developers,” Google. [Online]. Available: https://developers.google.com/machine-learning/recommendation/collaborative/summary.

[4] “Why and how to use Google Colab,” TechTarget, 2023 [Online]. Available: https://www.techtarget.com/searchenterpriseai/tutorial/Why-and-how-to-use-Google-Colab#:~:text=The%20benefits%20of%20Google%20Colab,or%20HTML%20as%20they%20go.

[5] “Python,” python. [Online]. Available: https://www.python.org/.

[6] “Pandas,” pandas. [Online]. Available: https://pandas.pydata.org/.

[7] NumPy. [Online]. Available: https://numpy.org/.

[8] “TensorFlow,” TensorFlow. [Online]. Available: https://www.tensorflow.org/?hl=tr.

[9] Victor. "Collaborative Filtering using Deep Neural Networks (in Tensorflow)", Medium, 20 Jun. 2019. Available: https://medium.com/@victorkohler/collaborative-filtering-using-deep-neural-networks-in-tensorflow-96e5d41a39a1.

[10] Kubler, Robert. “Introduction to Embedding-Based Recommender Systems”, Medium, 25, Jan. 2023. Available: https://medium.com/towards-data-science/introduction-to-embedding-based-recommender-systems-956faceb1919.

[11] “Scikit-learn,” scikit-learn. [Online]. Available: https://scikit-learn.org/stable/.

[12] Dhuriya, Ankur. “How to do a Content-Based Filtering using TF-IDF?”, Medium, 10, Nov. 2020. Available: https://medium.com/analytics-vidhya/how-to-do-a-content-based-filtering-using-tf-idf-f623487ed0fd.

[13] Folkman,Tyler. “Why You Are Using t-SNE Wrong”, Medium, 01, Dec. 2019. Available: https://towardsdatascience.com/why-you-are-using-t-sne-wrong-502412aab0c0.