National University of Computer & Emerging Sciences
Karachi Campus

Deep Learning for Perception-8A

Project Report

Book Recommendation System

20K-0265 Insia Farhan

20K-0409 Mukand Krishna

20K-0270 Faris Bin Asif

Objectives:

The project aims to create a sophisticated book recommendation system using collaborative filtering, Neural Collaborative Filtering, and content-based filtering techniques. By analyzing user ratings and preferences, personalized recommendations will be provided. Performance will be evaluated rigorously, considering metrics mean square errors, to ensure an improved user experience.

Problem Statement:

Developing an effective book recommendation system that addresses user preferences and improves the browsing experience remains a challenge. This project aims to integrate collaborative filtering, Neural Collaborative Filtering (NCF), and content-based filtering techniques to provide accurate and personalized book recommendations. By leveraging these methods, the system seeks to overcome the limitations of traditional approaches and deliver tailored suggestions that enhance user satisfaction and engagement.

Methodology:

1. **<u>Collaborative Filtering:</u>**
    a. **Data Preparation:**
        i. Import necessary libraries and load datasets (Books, Users, Ratings).
        ii. Clean data by handling missing values and dropping unnecessary columns.
    b. **Filtering and Matrix Creation:**
        i. Filter users with significant ratings and retain popular books based on rating counts.
        ii. Build a user-book rating matrix, filling NaN values with zeros.
    c. **Similarity Calculation:**
        i. Compute cosine similarity between books to create a similarity matrix.
    d. **Content-Based Recommendation:**
        i. Implement a function to recommend similar books based on a given book title.
    e. **Collaborative Filtering with Surprise:**
        i. Utilize Surprise library for collaborative filtering.
        ii. Train SVD model on training data, evaluate using RMSE on test set.
    f. **User-Based Recommendation:**
        i. Create a function to recommend books for a user based on collaborative filtering predictions.
    g. **Execution and Display:**
        i. Execute the code to provide recommendations for a specific user.
        ii. Display top recommended books with similarity scores.

2. Neural Collaborative Filtering (NCF):

- **a. Data Loading and Preparation:**
  - i. Load the book data, presumably containing user-item interactions (e.g., ratings).
  - ii. Rename columns to conform to LibRecommender's requirements.
- **b. Dataset Splitting:**
  - i. Split the dataset into training, evaluation, and test sets using random_split function.
- **c. Dataset Formatting:**
  - i. Format the datasets into LibRecommender's required format using DatasetPure.build_trainset, DatasetPure.build_evalset, and DatasetPure.build_testset.
- **d. Model Configuration:**
  - i. Configure the NCF (Neural Collaborative Filtering) model.
  - ii. Specify task type (rating), loss function (cross_entropy), embedding size, number of epochs, learning rate, batch size, and number of negative samples.
- **e. Model Training:**
  - i. Train the NCF model on the training data.
  - ii. Monitor metrics (e.g., loss) on the evaluation data during training.
- **f. Model Evaluation:**
  - i. Evaluate the trained model on the test data.
  - ii. Calculate and display evaluation metrics (e.g., loss).
- **g. Recommendation Generation:**
  - i. Generate top-N recommendations for a specific user using the trained model.
- **h. Display Results:**
  - i. Print the titles of recommended items for a user.
  - ii. Print some unique user IDs from the dataset.

3. Content-Based Filtering:

- **a. Data Loading and Exploration:**
  - i. Load the dataset containing books, users, and ratings data.
  - ii. Check for any null values in the book dataset and handle them if present.
  - iii. Merge the ratings data with cleaned books and user data to create a unified dataset.
- **b. Feature Engineering:**
  - i. Create a new feature by concatenating relevant columns (Book Title, Author, Publisher) to form a textual description of each book.
- **c. Data Cleaning and Preprocessing:**
  - i. Drop unnecessary columns.
  - ii. Convert the 'Features' column to lowercase to ensure consistency.

d. **Duplicate Handling:**
    i. Identify and handle duplicate entries in the dataset if any.
e. **Data Sampling:**
    i. Optionally, reduce the size of the dataset for testing purposes.
f. **Text Vectorization:**
    i. Utilize CountVectorizer to convert textual features into numerical vectors.
g. **Similarity Calculation:**
    i. Calculate cosine similarity between item profiles (book descriptions) to find similar items.
h. **Recommendation Generation:**
    i. Implement a function to recommend similar books based on a given book title.
    ii. Optionally, display the top recommended books along with their similarity scores.
i. **Duplicate Analysis:**
    i. Analyze and handle duplicate books in the sampled dataset if present.
j. **Summary Statistics:**
    i. Display the number of unique books and the shape of the sampled dataset.

4. GNN:

a. **Data Loading and Preprocessing:**
    i. Load the dataset, in this case, the books datasets (Books, Users, Ratings).
    ii. Normalize features if required.
b. **Graph Neural Network (GNN) Model Definition:**
    i. Define the GNN architecture, including the number of layers, hidden channels, and output classes.
    ii. Implement the forward pass method to define the computation flow.
c. **Model Training:**
    i. Train the GNN model.
    ii. Define the loss function (e.g., BPR loss).
    iii. Train the model over multiple epochs, optimizing the loss function.
d. **Model Evaluation:**
    i. Evaluate the trained model to measure its performance.
    ii. Calculate metrics such as loss to assess the model's effectiveness.
e. **Prediction:**
    i. Use the trained model to make predictions.
    ii. Generate recommended books for a subset of users based on their interactions.

Results:

1. Collaborative Filtering:

```
user_id = 271705
recommended_books = recommend_books(user_id)
print(f"Top {len(recommended_books)} recommended books for user {user_id}:")
for i, (title, similarity_score) in enumerate(recommended_books, start=1):
    print(f"{i}. {title} (Similarity Score: {similarity_score})")
```

```
Top 10 recommended books for user 271705:
1. Congo (Similarity Score: 8.137327419740483)
2. Tuesdays with Morrie: An Old Man, a Young Man, and Life's Greatest Lesson (Similarity Score: 6.725120979499343)
3. Naked (Similarity Score: 6.436396155553435)
4. Tears of the Moon (Irish Trilogy) (Similarity Score: 6.349517287773477)
5. Pride and Prejudice (Similarity Score: 6.246561368513779)
6. Brave New World (Similarity Score: 6.225049924842049)
7. Outlander (Similarity Score: 6.163086027357083)
8. One for the Money (Stephanie Plum Novels (Paperback)) (Similarity Score: 6.084350284049542)
9. Anne of Green Gables (Anne of Green Gables Novels (Paperback)) (Similarity Score: 6.078264417330809)
10. Shopaholic Takes Manhattan (Summer Display Opportunity) (Similarity Score: 6.043269244872453)
```

2. Neural Collaborative Filtering (NCF):

```
top_10_item_ids = ncf.recommend_user(user=14, n_rec=10)

top_3_titles = new_data[new_data['item'].isin(top_10_item_ids[14])]['Book-Title'].values[:6]
print(top_3_titles)
```

```
['Thing of Beauty' 'Hannibal'
 "Kingmaker's Sword (The Rune Blade Trilogy, Book 1)" 'Southern Cross'
 'Ein gutes Jahr.'
 "The White Dog Cafe Cookbook: Recipes and Tales of Adventure from Philadelphia's Revolutionary Restaurant"]
```

3. Content-Based Filtering:

```
def recommend_cv(book):
    index = np.where(content_cv['Book-Title'] == book)[0][0]
    similar_books = sorted(enumerate(cosine_sim[index]), key=lambda x: x[1], reverse=True)[1:10]

    print("\nTop 5 recommended movies based on your preferences:")
    for i in similar_books:
#         print(content_cv['Book-Title'][i[0]])
        print(content['Book-Title'][i[0]])
```

```
[ ]  recommend_cv('Chain of Evidence')
```

```
Top 5 recommended movies based on your preferences:
La Casa De Bernarda Alba
Die MÃ?Â¤dchen mit den dunklen Augen.
Maudit Manege
Rolling Thunder
Night Sins
A Kiss of Shadows (Meredith Gentry Novels (Paperback))
Disney's The Lion King (A Golden Look-Look Book)
Saving Private Ryan
Flesh Tones: A Novel
```

```
from tabulate import tabulate

def recommend_book_cv(book):
    index = np.where(content_cv['Book-Title'] == book)[0][0]
    similar_books = sorted(enumerate(cosine_sim[index]), key=lambda x: x[1], reverse=True)[1:6]

    table_data = []
    for i in similar_books:
        recommended_index = i[0]
#         recommended_title = content_cv['Book-Title'][recommended_index]
        recommended_title = content['Book-Title'][recommended_index]
        similarity_score = i[1]
        table_data.append([recommended_index, recommended_title, similarity_score])

    print(tabulate(table_data, headers=['Index', 'Recommended Book', 'Similarity Score'], tablefmt='grid'))
```

```
[ ]  recommend_book_cv('The Client')
```

Index	Recommended Book	Similarity Score
973	Witch's Christmas	1
156	Auf dÃ?Â¼nnem Eis.	0.912871
941	An ALTOGETHER NEW BOOK OF TOP TEN LISTS LATE NIGHT DAVID LETTERMAN	0.833333
351	Contacto	0.612372
916	Ordinary Love and Good Will	0.547723

```
recommend_book_cv('Deadly Decisions')
```

Index	Recommended Book	Similarity Score
304	The Coral Island (Puffin Classics)	0.67082
471	El espectador	0.353553
22	Reise nach Ixtlan. Die Lehre des Don Juan.	0.288675
815	The Cat Who Robbed a Bank (Cat Who... (Paperback))	0.288675
385	Dawn and the We Love Kids Club (Baby-Sitters Club, 72)	0.223607

```
recommend_book_cv('Bright Eyes (Coulter Family Series)')
```

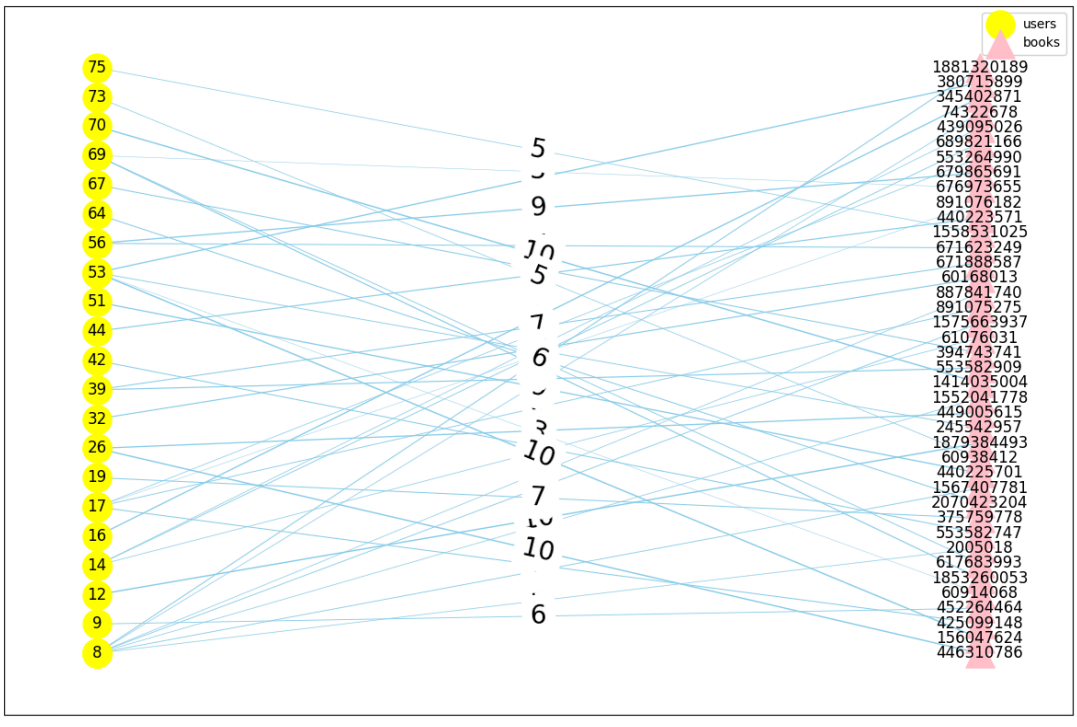
Index	Recommended Book	Similarity Score
957	Something Special	1
190	O Pioneers! (Bantam Classic)	0.5
654	Mother Night	0.5
409	Cosmetique De L'Enneme	0.408248
681	Amazonia	0.408248

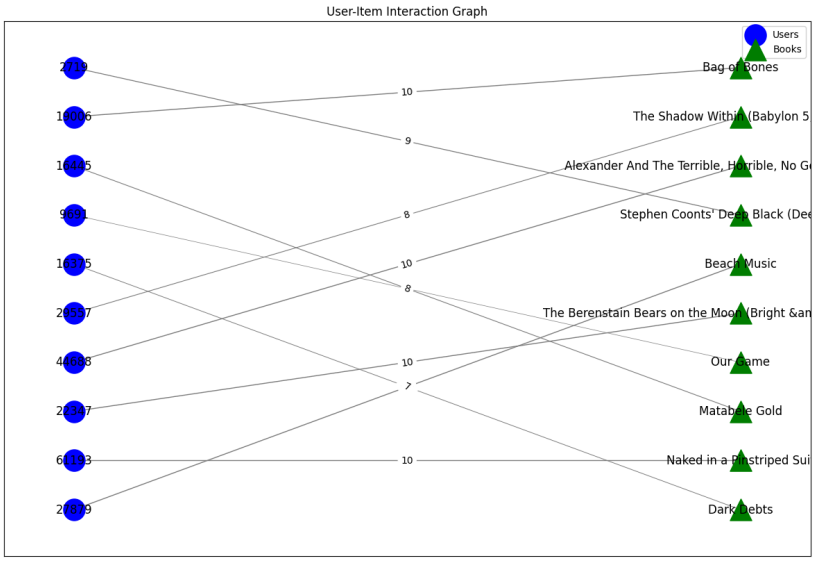
4. Graph Neural Network:

Graph Neural Networks in recommendation systems typically run on collaborative filtering data. Collaborative filtering leverages user-item interactions to make recommendations. These interactions can be explicit, such as ratings or reviews, etc.

In the context of GNNs, collaborative filtering is represented as a **bipartite graph**, where nodes represent users and items, and edges represent interactions between them. GNNs learn to propagate information through this graph structure to capture user-item relationships and make personalized recommendations.

Tried multiple approaches to provide recommendations using GNN. And created plots of user and item interactions.





However, due to multiple errors mainly index errors using tensors, couldn't generate recommendations.

```
No recommendations found for user 10
No recommendations found for user 11
No recommendations found for user 12
No recommendations found for user 13
No recommendations found for user 14
No recommendations found for user 15
No recommendations found for user 16
No recommendations found for user 17
No recommendations found for user 18
No recommendations found for user 19
```

References:

1. Python Awesome (GNN)
2. Collaborative Filtering using DNN
3. Light GCN Self-Supervised Learning
4. Graph Neural Networks
5. Recommendation System using NCF