AIE425 Intelligent Recommender Systems, Fall Semester 24/25

Assignment #1: Neighborhood CF Models (User, Item-Based CF)

ID:222102535, Karim Mamdouh Ahmed

Exploration of Amazon and LinkedIn Recommender Systems

Recommender systems are central to digital platforms, helping users discover products, content, or connections that align with their preferences. Here, I explore how Amazon and LinkedIn use recommender systems, the types of customer feedback they collect, and the methods for analyzing this data.

Amazon's Recommender System

Amazon, as a major e-commerce platform, relies heavily on its recommendation system to drive product discovery and sales. Amazon's system uses both explicit and implicit feedback from users to make recommendations.

Customer feedback is collected through star ratings (1-5 stars) and written reviews, providing direct insights into customer satisfaction with specific products. Additionally, Amazon allows users to mark reviews as helpful, which further refines its recommendation accuracy. Beyond explicit ratings, Amazon gathers implicit feedback from user actions such as browsing history, purchase history, time spent on product pages, and items added to the wishlist. This data helps Amazon gauge user interest without requiring direct input.

Amazon's recommendation system combines collaborative filtering (mainly item-based), content-based filtering, and deep learning techniques. For example, the "Customers who bought this also bought" feature is powered by item-based collaborative filtering, while deep learning models create more personalized suggestions on users' homepages.

LinkedIn's Recommender System

LinkedIn uses its recommendation system to help users connect with other professionals, find job opportunities, and engage with relevant content. Since explicit feedback on connections or job opportunities is rare, LinkedIn focuses more on implicit signals.

LinkedIn gathers implicit data based on user actions such as profile views, connection requests, endorsements, and job applications. This data helps identify a user's professional interests and recommend suitable connections or job listings. Additionally, LinkedIn allows users to endorse each other's skills, which serves as a form of feedback. Engagement metrics like likes, comments, and shares on posts provide LinkedIn with insights into content relevance.

LinkedIn's recommendations rely on graph-based approaches, like the "Random Walk with Restarts" algorithm, for suggesting connections (e.g., "People You May Know"). For job recommendations and content suggestions, LinkedIn employs collaborative filtering, natural language processing, and graph neural networks to refine recommendations.

Background on User-Based and Item-Based Collaborative Filtering

Collaborative Filtering (CF) is a popular approach in recommender systems for predicting a user's interest in an item based on the interests of similar users or items. CF algorithms are primarily divided into two types: **User-Based Collaborative Filtering** and **Item-Based Collaborative Filtering**.

User-Based Collaborative Filtering

User-Based Collaborative Filtering operates on the assumption that users with similar past preferences will also have similar future preferences. This method identifies users who have rated items similarly to the target user and uses their ratings to predict the target user's potential interests.

The first step in User-Based CF is calculating user similarity, which can be done using metrics like cosine similarity or Pearson correlation. For users u and v with ratings r_{ui} and r_{vi} on item i, the similarity $\text{sim}(u, v)$ is calculated as:

$$\text{sim}(u, v) = \frac{\sum_{i \in I}(r_{ui} - \bar{r}_u)(r_{vi} - \bar{r}_v)}{\sqrt{\sum_{i \in I}(r_{ui} - \bar{r}_u)^2}\sqrt{\sum_{i \in I}(r_{vi} - \bar{r}_v)^2}}$$

where I is the set of items rated by both users, and $\bar{r}_u$ and $\bar{r}_v$ are the average ratings of users u and v.

For predicting a user u's rating $\hat{r}_{ui}$ on item i, a weighted average of ratings from similar users is used:

$$\hat{r}_{ui} = \bar{r}_u + \frac{\sum_{v \in N_u} \text{sim}(u,v)(r_{vi} - \bar{r}_v)}{\sum_{v \in N_u} |\text{sim}(u,v)|}$$

where N_u represents users similar to u.

Item-Based Collaborative Filtering

Item-Based Collaborative Filtering assumes that similar items will receive similar ratings from users. This model identifies items that are similar to the items a user has liked and recommends them to the user.

The first step in Item-Based CF is calculating item similarity, using measures like cosine similarity or Pearson correlation. For items i and j rated by users u, the similarity $\text{sim}(i,j)$ is calculated as:

$$\text{sim}(i,j) = \frac{\sum_{u \in U}(r_{ui} - \bar{r}_i)(r_{uj} - \bar{r}_j)}{\sqrt{\sum_{u \in U}(r_{ui} - \bar{r}_i)^2}\sqrt{\sum_{u \in U}(r_{uj} - \bar{r}_j)^2}}$$

where U is the set of users who rated both items.

For predictions for a user u on an item i, the calculation is:

$$\hat{r}_{ui} = \bar{r}_i + \frac{\sum_{j \in N_i} \text{sim}(i,j)(r_{uj} - \bar{r}_j)}{\sum_{j \in N_i} |\text{sim}(i,j)|}$$

where N_i represents items similar to i that u has rated.

Data Preparation and Preprocessing Steps

The dataset shown in Table 1 was collected through web scraping (using Python) from Amazon book review pages. The ratings use a 5-star scale, with 5 indicating the highest satisfaction.

I gathered reviews across 20 books, but most customers reviewed only one book, which limited overlap in the dataset. After analyzing the reviews, I found that only one customer provided multiple reviews across different books, contributing three unique ratings.

To prepare the data for analysis, I performed the following preprocessing steps:

1.Selection of Customers and Books: I selected five representative customers and four books for inclusion in the table. This selection process ensured that the data was manageable while still providing useful insights.

2.Handling Missing Values: For the customers who only had one review, I randomly filled the missing values to ensure that each customer had at least one missing entry, except for the customer who had three authentic reviews. This approach allowed for a more balanced dataset while maintaining the integrity of the reviews.

3.Data Extraction Techniques: I used Python libraries such as BeautifulSoup and Requests for HTML parsing and data extraction. These libraries allowed me to navigate the structure of the Amazon review pages efficiently and collect the relevant data. Additionally, I employed the CSV library to store the data.

Customer Name	**Book 1**	**Book 2**	**Book 3**	**Book 4**	**Book 5**
Laura M	5	3		5	
Kindle Customer MJ	4	2	1	4	1
HuntleyMC	3	1	3	5	3
The Littles	4		2	4	1
Markus Robbins	3		3	4	2
Bruce_in_LA	4	2		5	
E. Doty	3			4	3
A. T. Young	2	3	4	2	

Table 1: Sample Ratings from Selected Customers

Results Comparison

Title	**Cosine Similarity**	**Pearson Correlation**	**Adjustment**
Atomic	2.49	2.37	6.57
Framed	2.69	2.41	1.487053
Jobs	4.10	3.97	10.196838

Table 2: Comparison of Predicted Ratings Using Cosine Similarity, Pearson Correlation Coefficient, and Adjustment Values

Analysis of Results

Cosine Similarity

- For the book *Atomic Habits*, the predicted rating using cosine similarity is 2.49. - *Framed* received a rating prediction of 2.69 using cosine similarity. - For *Jobs to Be Done*, cosine similarity gave a significantly higher predicted rating of 4.10.

These values reflect how cosine similarity leverages the direction of rating patterns without centering around each user's or item's average. This often results in predictions that emphasize general trends, especially if there is a consistent rating pattern.

Pearson Correlation

- For *Atomic Habits*, the Pearson correlation-based prediction is 2.37, slightly lower than cosine similarity. - *Framed* received a lower predicted rating of 2.41 with Pearson, compared to cosine similarity. - There is no predicted rating for *Jobs to Be Done* using Pearson correlation, likely due to sparse data or a lack of user overlap.

Pearson correlation considers individual rating biases by centering on each user's or item's mean rating. Consequently, it captures the relative rating tendency and is generally more responsive to users who deviate from the average.

Pros and Cons of Each Technique

Similarity Measure	**Advantages**	**Disadvantages**
Cosine Similarity	- Scale-invariant, emphasizing overall pattern of ratings - Computationally efficient	- Ignores individual rating biases - Less effective with sparse data and limited overlap
Pearson Correlation Coefficient	- Adjusts for rating biases, capturing relative preferences - Effective when individual rating scales differ	- Sensitive to outliers - Higher computational cost, especially with sparse data

Table 3: Pros and Cons of Cosine Similarity and Pearson Correlation Coefficient

Customer Name	**Average Rating**
Laura M	3.31
Kindle Customer MJ	2.89
HuntleyMC	2.68
The Littles	3.35
Markus Robbins	3.28
Bruce_in_LA	3.44
DAI	4.05
Heather Key	3.42
Kristen Stoeger	2.74
Nick Hutchison	3.42
Debs Stuff	2.89

Table 4: Average Ratings for Each Customer

Implementation Process

The implementation process for this project involved several stages: web scraping, data preprocessing, similarity computation, and rating prediction. First, we used `BeautifulSoup` and `requests` to scrape book reviews from Amazon. This data included customer names, book titles, and ratings, which were organized into a structured format using `pandas`.

To preprocess the data, we handled missing values by filling them with random ratings for testing purposes and calculated average ratings for each customer. For similarity computations, we utilized `scipy.spatial.distance` for cosine similarity and `scipy.stats.pearsonr` for Pearson correlation. These metrics were then applied in both user-based and item-based collaborative filtering models, where each user or item's similarity was calculated to predict missing ratings.

Tools and Libraries:

- `BeautifulSoup` and `requests` for web scraping
- `pandas` for data manipulation and preprocessing
- `scipy` for similarity computations
- `numpy` for mathematical operations and handling missing values

User vs Item-Based CF

User-based and item-based collaborative filtering (CF) approaches have distinct characteristics that impact prediction accuracy and recommendation quality:

- **User-Based CF**: This approach finds similarities between users by comparing their ratings on common items. Using cosine similarity, we focus on the "angle" of their preferences, while Pearson correlation evaluates the linear relationship, accounting for the user's average rating. In practice, user-based CF can provide personalized recommendations but often suffers from sparsity issues when users rate few items in common.

- **Item-Based CF**: In item-based CF, similarities are calculated between items, focusing on how frequently users rate two items similarly. Item-based CF can be more stable as items are typically rated by more users than users rate items, reducing sparsity. Adjusted cosine similarity, which considers users' average ratings, often provides more accurate similarity scores between items. Pearson correlation can also reveal items with similar rating patterns but may be less robust when the number of common ratings is low.

Remarks on Similarity Measures: - **Cosine Similarity**: Effective in capturing overall similarity patterns but may be less sensitive to individual rating tendencies. Cosine similarity performs well when the focus is on rating direction rather than rating levels. - **Pearson Correlation**: More nuanced, as it adjusts for individual rating biases, making it well-suited for applications where rating scales vary among users. However, it may be less stable if few ratings overlap.

Conclusion

Each collaborative filtering strategy influenced predicted accuracy in unique ways:

- **User-Based CF**: The accuracy of predictions depended heavily on the availability of sufficient overlapping ratings between users. For sparse datasets, Pearson correlation provided more accurate predictions by adjusting for rating scale differences, but it required a minimum overlap in ratings for meaningful results. Cosine similarity, while straightforward, sometimes provided less accurate predictions due to its lack of adjustment for user-specific rating tendencies.

- **Item-Based CF**: Item-based CF, particularly with adjusted cosine similarity, generally offered more accurate predictions as it accounted for user rating biases and leveraged the relatively high number of users rating each item. Adjusted cosine similarity captured similarity between items effectively, leading to more reliable predictions. Pearson correlation also worked well here but occasionally struggled with items that had limited shared ratings.

In summary, item-based CF typically provided more stable and accurate predictions than user-based CF in this dataset, as the similarity calculations between items were more robust to sparsity. Adjusted cosine similarity emerged as a reliable measure for capturing item relationships, while Pearson correlation's accuracy varied depending on the overlap of ratings.

Enhancements

There are several enhancements that could improve recommendation accuracy and system performance:

1. **Incorporating Hybrid Models**: Combining user-based and item-based CF methods could leverage the strengths of each approach. By integrating both models, we can create a more robust system that accounts for both user preferences and item similarities, particularly useful when dealing with sparse data.

2. **Use of Matrix Factorization**: Techniques like Singular Value Decomposition (SVD) or Alternating Least Squares (ALS) could enhance recommendation quality by uncovering latent features in the user-item matrix. These methods can capture more nuanced relationships between users and items, which is beneficial for prediction accuracy.

3. **Incorporating Implicit Feedback**: Many users may browse or partially read books without leaving a rating. Incorporating implicit feedback (e.g., time spent on pages, clicks) could provide additional insights into user preferences, enriching the dataset and improving model performance.

4. **Data Imputation Techniques**: Filling in missing values with more sophisticated techniques, such as K-nearest neighbors imputation or model-based predictions, could reduce sparsity and enhance similarity calculations. This approach would provide more reliable similarity scores for both user-based and item-based CF.

5. **Exploring More Advanced Similarity Metrics**: Additional similarity measures, such as Jaccard similarity for binary data (liked/disliked) or Spearman correlation for ranking-based data, could offer alternative insights into user and item relationships, potentially improving recommendation diversity.

Each of these enhancements could further improve the robustness and accuracy of the recommendation system, creating a more user-centric and effective model.