Homework 1: News Feed Recommendation

Shi Qiu

The George Washington University

Prof. Sardar Hamidian & Prof. Armin Mehrabian

Abstract—Apply your collaborative filtering approach based on what we covered in class. You can use content-based, user-based, matrix factorization (MF), or even come up with your own idea. You can implement conventional methods, clustering, SVD, etc. Note that it doesn't have to involve deep learning.

Keywords—*Recommended System, collaborative filtering, content-based, user-based, SVD*

1. Introduction

In Homework 1, we utilize the MIND dataset, which contains 156,965 interactions, 50,000 news articles, and 50,000 users. For my approach, I focus primarily on the subcategory of the news in both content-based and user-based collaborative filtering methods. Additionally, I implement SVD as a third approach for matrix factorization. To evaluate the results, I measure the performance using recall and precision across all methods.

2. Data Preprocessing

The dataset used for this project is the MIND-small dataset, which primarily consists of two TSV files: `behaviors.tsv` and `news.tsv`. The `behaviors.tsv` file contains records of user interactions with news articles, including fields like `impressionId`, `userId`, `timestamp`, `click_history`, and `impressions` (detailing clicked and not-clicked news items). The `news.tsv` file provides metadata about the news articles, such as `itemId`, `category`, `subcategory`, `title`, `abstract`, `url`, `title_entities`, and `abstract_entities`.

2.1. News Data Processing

I removed the `category` field since the focus of the analysis is on the `subcategory` field. Additionally, I dropped other unnecessary fields such as `title`, `abstract`, `url`, `title_entities`, and `abstract_entities`. Next, I created two dictionaries: one mapping subcategory IDs to news IDs and another mapping news IDs to subcategory IDs, to facilitate efficient lookups between news articles and their corresponding subcategories.

2.2. Behaviors Data Processing

For the behaviors data, the `impressions` field contains both clicked and not-clicked information in the form of `"N55689-1 N35729-0"`. I processed this field by splitting the entries and distinguishing between `1` (clicked) and `0` (not clicked). The `userId` field is used to identify each unique user, allowing their interaction history to be aggregated. Since the `timestamp` field was not required for this analysis, it was dropped.

I also grouped the processed data into a format containing only `userId`, `history`, `clicks`, and `noclicks`, summarizing each user's interactions with the news articles.

3. Approach

3.1. User-Based CF

In the User-Based Collaborative Filtering approach, I compute each user's preference scores based on their browsing history and generate recommendations from similar users' histories. The process is as follows:

1. **Calculating User Preference Scores:** For each user, I analyze their browsing history and count the number of news articles they read in each subcategory to build a subcategory preference profile. I then apply the softmax function to normalize these preferences.
2. **Computing User Similarities:** Using the normalized preference scores, I calculate cosine similarity between users to identify the top-k most similar users.
3. **Generating Recommendations:** I gather news articles from the browsing histories of the most similar users. Each news article is weighted based on the similarity score of the corresponding user. The recommendation list is ranked by weighted scores and provided to the user.

3.2. Content-Based Collaborative Filtering

In the Content-Based Collaborative Filtering approach, similar to the user-based method, I use each user's browsing history to identify the subcategories of the news articles they have read. I then apply the softmax function to these subcategory counts to generate preference scores for each user. Based on these scores, I recommend news articles that belong to the subcategories the user has shown the most interest in.

3.3. Singular Value Decomposition (SVD)

For the SVD approach, I build a large user-news interaction matrix where rows represent users and columns represent news articles. If a user clicked on a news article, the corresponding entry in the matrix is set to 1; otherwise 0.

I use built-in library functions to perform Singular Value Decomposition (SVD) on this sparse matrix. Due to memory constraints, I lower the precision to `float32` for efficient processing. Initially, I set the number of latent factors $k = 50$ and experiment with various k values to observe their impact on recommendation performance.

4. Experimental Setup and Evaluation Metrics

For this experiment, I used users' browsing history, specifically the `click_history` field from `behaviors.tsv`, to calculate recommendations. This historical data served as the training set to compute each user's preference scores and generate personalized recommendations.

To evaluate the effectiveness of the recommendations, I compared them with the actual clicks recorded in the `impressions` field, where news articles were marked as clicked (`1`) or not clicked (`0`). This comparison allowed me to assess how well the recommended news matched the users' true interests.

The evaluation metrics used were:

- **Recall**: The proportion of relevant news successfully recommended out of all the articles the user clicked on. Higher recall means better coverage of the user's interests.
- **Precision**: The proportion of relevant news among all recommendations. Higher precision indicates fewer irrelevant recommendations.

This setup reflects real-world usage, where recommendations are based on past user behavior, and their effectiveness is measured by how well they match the user's actual clicks.

5. Results

The parameter *Top N* represents the number of news articles recommended to the user. For example, if *Top N* is 100, it means the system

recommended 100 news articles. A recall score of 0.4833 indicates that 48.33% of the actual news articles clicked by the user are included within these 100 recommended articles.

5.1. Evaluation of Different Recommendation Methods

Figure 1 and Table 1 show the Recall and Precision for different methods across various values of *Top N*. As we can see from the graph and the table, the user-based method exhibits the best performance among all values of *N*, while content-base and svd(k=50) did not make much difference

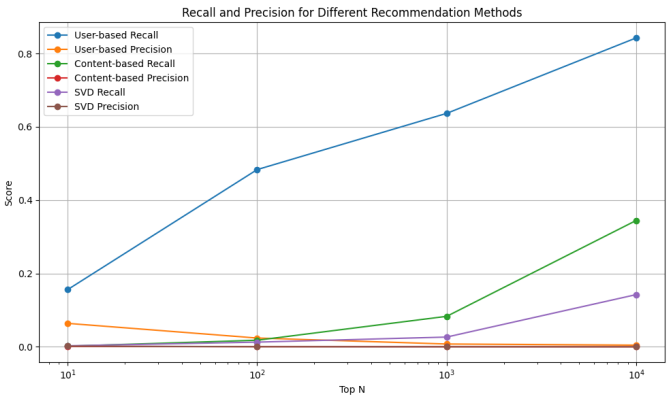


Figure 1. Recall and Precision for different methods across varying *Top N* values.

Table 1. Comparison of Recommendation Methods

Top N	User-based		Content-based		SVD (k=50)	
	Rec.	Prec.	Rec.	Prec.	Rec.	Prec.
10	0.1559	0.0640	0.0018	0.0010	0.0007	0.0010
100	0.4833	0.0240	0.0180	0.0006	0.0012	0.0002
1000	0.8422	0.0047	0.0832	0.0004	0.0267	0.0001
10000	0.8426	0.0046	0.3448	0.0002	0.1425	0.0001

User-based Collaborative Filtering: The user-based method achieves the highest recall and precision across all values of *Top N*. This is because the user-based approach effectively leverages the browsing history of other similar users. By using collaborative filtering, the model benefits from real user behavior, where similar users manually select relevant news. This leads to more accurate and personalized recommendations. As *Top N* increases, recall improves significantly, indicating that more relevant news is being recommended.

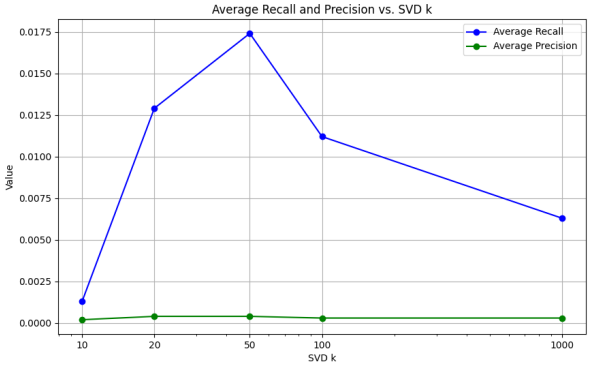
Content-based Filtering: In contrast, the content-based did not perform well. It recommended news articles from the user's preferred subcategory, so most recommendations come from the same subcategory, which lead the low recall and precision.

SVD-based Recommendations: The SVD method performs the weakest among the three approaches. The poor performance is likely due to the high sparsity of the interaction matrix, where users interact with only a small subset of the total news collection. As a result, the SVD model struggles to extract meaningful latent factors, leading to low recall and precision scores.

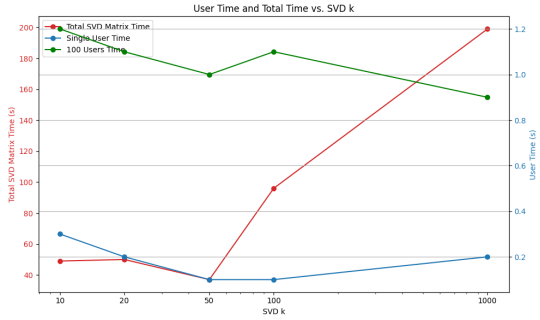
However, as shown in Table 2 and Figure 2, we observe that after the time-consuming computation of the SVD matrix, the time taken to generate recommendations for each user is significantly reduced. This advantage can be leveraged for pre-computation, which may help minimize the waiting time when users refresh the page. For instance, after the SVD matrix is computed, generating recommendations for 100 users only takes 1.2 seconds, whereas the user-based approach takes 16 seconds for the same number of users.

5.2. SVD Performance Analysis

Figure 2 and Table 2 illustrate the performance of the SVD method with different values of the latent factor k.



(a) Recall and Precision scores for SVD with varying k.



(b) Computation time for SVD with varying k.

Figure 2. Performance of the SVD method with different k values.

From Table 2 and Figure 2, we can observe how different values of k affect the performance of the SVD-based recommendation method. Generally, increasing k allows the model to capture more latent dimensions, preserving more meaningful patterns in the data. However, this comes at the cost of increased computation time, as shown by the rise in time as k increases from 10 to 50.

Table 2. SVD Performance with Different k Values

k	SVD Time	1 User Time	100 Users Time	Rec.	Prec.
10	49	0.3	1.2	0.0013	0.0002
20	50	0.2	1.1	0.0129	0.0004
50	37	0.1	1.0	0.0174	0.0004
100	96	0.1	1.1	0.0112	0.0003
1000	199	0.2	0.9	0.0063	0.0003

Interestingly, while the recall improves significantly as k increases up to 50, the performance starts to drop when k exceeds 50. This decline in performance is likely due to overfitting, as the model begins to capture noise rather than meaningful relationships within the data. Furthermore, the high sparsity of the interaction matrix means that the model struggles to identify relevant factors at larger values of k, leading to diminishing returns in recommendation quality.

6. FYI

It's important to note that the click data used for evaluation comes from news already recommended by Microsoft's system. This may skew the results, as users were limited to the articles presented by the existing recommendation engine, potentially misleading the performance metrics of the models tested.

7. References

1. Collaborative Filtering: Data Science Concepts. (n.d.). YouTube. Retrieved from https://www.youtube.com/watch?v=Fmtorg_dmM0
2. MIND: Microsoft News Recommendation Dataset. (n.d.). Kaggle. Retrieved from https://www.kaggle.com/datasets/arashnic/mind-news-dataset/code?datasetId=1049650&sortBy=voteCount
3. MIND Recommender from Scratch. (n.d.). Kaggle. Retrieved from https://www.kaggle.com/code/enemis/mind-recommender-from-scratch/notebook