

A Novel Approach to Recommendation System Business Workflows: A Case Study for Book E-Commerce Websites

Mounes Zaval(✉), Said Orfan Haidari, Pinar Kosan, and Mehmet S. Aktas

Computer Engineering Department, Yildiz Technical University, Istanbul, Turkey
{mounes.zaval,said.haidari,pinar.kosan}@std.yildiz.edu.tr,
aktas@yildiz.edu.tr

Abstract. Have you ever wondered why a song or a book or a movie becomes so popular that everyone everywhere starts talking about it? If we did not have the technology, we would say that people who love something would start recommending it to their friends and families. We live in the age of technology where there are so many algorithms that can discover the patterns of human interaction and make an excellent guess about someone's opinion about something. These algorithms are building blocks of digital streaming services and E-Commerce websites. These services require as accurate as possible recommendation systems for them to function. While many businesses prefer one type or another of recommendation algorithms, in this study, we developed a hybrid recommendation system for a book E-Commerce website by integrating many popular classical and Deep Neural Network-based recommendation algorithms. Since explicit feedback is unavailable most of the time, all our implementations are on implicit binary feedback. The four algorithms that we were concerned about in this study were the well-known Collaborative filtering algorithms, item-based CF and user-based CF, ALS Matrix Factorization, and Deep Neural Network Based approaches. Consequently, comparing their performances and accuracy, it was not surprising that the Deep Neural Network approach was the most accurate recommender for our E-Commerce website.

Keywords: E-commerce · Deep neural network · Collaborative filtering · Alternating least square · Matrix factorization

1 Introduction

Recommendation engines perform a critical function in mitigating existing overgrowing information overload. Hence, they have been utilized and used by several platforms in many different ways, such as recommending a video on Youtube, a friend on Facebook, a movie on Netflix, or a product to purchase on E-commerce sites.

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2022
O. Gervasi et al. (Eds.): ICCSA 2022 Workshops, LNCS 13381, pp. 692–708, 2022.
https://doi.org/10.1007/978-3-031-10548-7_50

One of the most significant barriers to implementing recommendation engines for E-commerce sites is the lack of one user's interactions with the site products-leading to serious difficulties in finding sufficient reliable similar users and the problem of a high data sparsity rate. Hence, in this study, we try to solve the long tail problem and propose a better approach to make better recommendations for less popular items. Traditional recommendation systems, including user-Based Collaborative Filtering, item-based collaborative filtering, and Matrix Factorization-Based recommendation engines, expect actual product ratings to produce good results and have shown poor results when making recommendations based on implicit feedback. Furthermore, they cannot efficiently model user-item interactions when the user-item matrix (Rating Matrix) is highly sparse.

Motivation: There is a need for business workflows for recommendation systems that can allow us to choose different recommendation algorithms as needed. In particular, there is a need for business workflows that could employ not only the classical recommendation algorithms but also deep neural network-based algorithms. Such recommendation system business workflows can be utilized on different E-commerce websites. We are motivated to use this business workflow for a book E-commerce website in this study. There are several methods to extract users' feedback from the website's log files, the most popular method is explicit feedback collecting_when users explicitly rate items_. However, in some cases, there is no explicit way of collecting users' feedback. Therefore, other feedback collecting methodologies like binary feedback (implicit feedback) _based on users' purchases or items click-through, Etc._ are also needed and need to be investigated.

Research Problem: In this paper, we are investigating a novel business workflow that can employ different recommendation algorithms and produce recommendations based on implicit feedback methodology.

Here we list the questions studied under this research problem:

1. What would be the abstract modular software architecture of recommending items on a book E-commerce website?
2. How implicit feedback collecting mechanism can be used in different recommendation algorithms?
3. Which one of the recommendation algorithms would perform better on binary feedback dataset collected from an E-commerce website?

Contribution: In this study, we propose a business workflow that can be used as a recommendation system on E-commerce websites. To understand the usefulness and the benefits of this workflow, we implemented a prototype and did an experimental study to determine the effectiveness of this prototype.

Our experimental results show that among the other recommendation algorithms deep neural network learning approach was the best performer providing better recommendations relatively.

In this study, we also investigated the use of the implicit feedback approach as a feedback collection mechanism; Our results show that this mechanism of gathering users' feedback is also able to produce good results.

Organization of the Paper: We will start by introducing the fundamental concepts a reader should be aware of and then talk about some similar studies in the literature review. Then we will dive in to explain the methodology followed throughout constructing the proposed business workflow architecture for a book E-commerce website's recommendation engine from data flow and modular perspectives. As we finish explaining the methodology, we clarify the used technologies throughout the implementation of the proposed architecture. Furthermore, we report the final performance results for each Recommendation algorithm in the experimental results sections.

2 Fundamental Concepts and Literature Review

2.1 Fundamental Concepts

Collaborative Filtering: The collaborative filtering (CF) technique is one of the most popular recommendation systems out there. CF addresses content-based recommendation's problem of keeping the recommendation in the same circle of what he likes and not trying to predict what the user might like based on other factors. The main idea of collaborative filtering is that two active users show similar behavior (e.g., click-through same items, rate items similarly). Therefore they most likely will share a similar opinion regarding other items [1].

For the sake of giving an example, we assume that a set of users user_1, user_2, user_3, user_1 and user_2 are the most similar and we want to predict what user_1 would rate a specific item (Y), then we seek help from user_2 in a way that reflects the most similar user's opinion of an item. Hence, if user_2 likes item (Y), then most likely our target user will like the item (Y) too, and vice versa [2].

Collaborative filtering systems encounter several challenges, especially when integrated with big E-commerce companies. Such as [3]:

- **Data Sparsity** Usually, the number of items immensely exceeds the number of users. As a result, a single user cannot interact with all items but with a small set of items, which brings about very high sparsity in the user-item matrix.
- **Scalability** The more users and items we have, the more expensive it becomes to produce CF-based recommendations. Thus, if the number of users and items increases dramatically, then CF systems will suffer from serious scalability issues [4].
- **Gray Sheep** Concerns those whose thoughts are not always on the same line as any other group of users, and hence they do not take advantage of Collaborative Filtering [5].

– **Others** like shilling attacks [6] and Synonymy [7].

For CF systems to provide high-quality recommendations, they must address these challenges in a productive and well-thought manner. The success of a CF system depends on the answer to the question: How **well** does the system handle the environmental challenges?

User-Based CF [8]**:** Following the same approach explained in the previous section, User-based CF calculates user similarity from a user-item matrix _which is a matrix that draws users' interactions with items_ and then generates recommendations based on this information. The used similarity metric is essential in the process of generating recommendations for active users. It must have some properties, such as considering the actual rating values and properly handling the missing ratings without treating them as negatives. Some well-known similarity metrics are [9]:

1. Pearson Correlation Coefficient Similarity.
2. Log-Likelihood Ratio Similarity.
3. Euclidean Distance Similarity.

And many others. As stated earlier, choosing the appropriate similarity measure is crucial when building a recommendation engine since each similarity metric has a different formula and yields different results. User-item matrix is constructed based on users' past experiences and interactions or ratings of items. User-item matrix is constructed based on users' past experiences and interactions or ratings of items as shown in Table 1.

Table 1. User-Item matrix (rating matrix)

Users\Items	$Item_1$	$Item_2$	$Item_3$
$User_1$	r_{11}		r_{13}
$User_2$	r_{21}	r_{22}	r_{23}
$User_3$		r_{32}	

Where: $r_{i,j} \rightarrow$ Rating of $item_j$ by $user_i$.

Item-Based CF [8]**:** Item-based collaborative filtering and user-Based collaborative filtering follow the same algorithm with slight differences. Item-based CF is concerned with generating recommendations for active users based on items' similarities rather than users' similarities. Hence, it applies the same calculations but on a different matrix used in user-based CF. Item-user matrix is constructed by taking the matrix transpose of the rating matrix. Item-user matrix is constructed by taking the matrix transpose of the rating matrix as shown in Table 2.

Table 2. Item-User matrix (rating matrix)

Items\Users	$User_1$	$User_2$	$User_3$
$Item_1$	r_{11}	r_{21}	
$Item_2$		r_{22}	r_{32}
$Item_3$	r_{13}	r_{23}	

Where: $r_{i,j} \rightarrow$ Rating of $item_j$ by $user_i$.

Even though item-based and user-based CF are the same theoretically, item-based CF has shown better and more realistic results outperforming user-based CF in the real world. Basically, this is due to users' opinions changing nature over time and having varied tastes. On the other hand, items belong to a small set of genres, and they are much more straightforward.

Matrix Factorization (MF): It has been proven empirically that the most powerful and helpful latent factor models are built on top of MF [10,11]. MF factorizes user-item interactions inferred from the rating matrix and represents these interactions in a shared latent factor space of N dimensions. Assume that p_u and $q_i \rightarrow User_u$ latent vector and $Item_i$ latent vector, then they would be represented in this joint latent factor space by p_u and q_i inner product. Hence, MF can estimate an interaction y_{ui} by[12]:

$$\hat{y_{ui}} = p_u^T q_u = \sum_{i=1}^{N} p_{un} q_{in} \tag{1}$$

where N is the dimension of the latent factor space. The matrix factorization approach introduces a well-thought solution to the scalability problem, which makes it preferred over other techniques when dealing with large datasets. Additionally, MF can handle and reduce the problem of high sparsity, especially when combined with singular value decomposition (SVD) or principal component analysis (PCA) to reduce dimensionality.

2.2 Literature Review

The attempt to design new hybrid recommendation systems has been studied over the years, as it improves the recommendation results and reduces the uninteresting recommendations. For instance, [13] in the "Hybrid recommender systems for electronic commerce" paper, a hybrid recommendation engine is investigated. They combined both the collaborative filtering recommendation algorithm and the knowledge-based recommendation algorithm _ [14] KBRS uses the acquired knowledge that is inferred from the database about users and items interactions to generate recommendations for the active users attempting to meet the user's requirements._ under one software architecture that provides a recommendation system scheme based on CFRS and KBRS. In 2000, "Hybrid Recommender

Systems: Survey and Experiments" has illustrated several hybridization methods for implementing a hybrid recommendation system [15], including:

1. **Weighted** The final score of a recommended item is calculated based on the recommendation algorithms votes present in the system for this item. The P-Tango system [5] uses such a hybridization approach.
2. **Mixed** Recommendations of all the participated recommendation algorithms are present at the same time. This method is used by the PTV system [16] to produce a recommended television viewing program.
3. **Feature combination** This method is used in integrating content-based and collaborative-based algorithms by considering collaborative information to be simply extra feature data and applying Content-Based approaches to process the enriched dataset [17].

We observe a number of studies in recommendation systems that utilize a variety of approaches, such as rule-based recommendation systems, case-based reasoning-based recommendation systems, and hybrid recommendations systems using collaborative filtering-based algorithms [18–24]. However, in this particular study, we propose a software architecture that follows a specific methodology to produce a successful recommendation system utilizing traditional and deep neural network-based algorithms for a book e-commerce website. There exist studies that focus on analyzing user's browsing behaviour (i.e. click-stream based browsing graph data) to extract knowledge [25–28]. In this study, we focus on user's purchase behaviour on the book e-commerce website.

3 Methodology

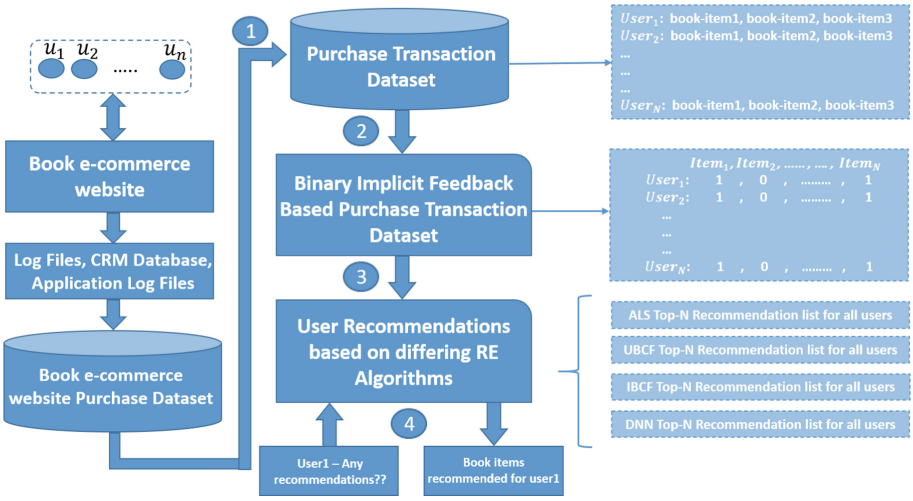

Fig. 1. The proposed business workflow architecture for a book E-commerce website Recommendation Engine _ Data Flow

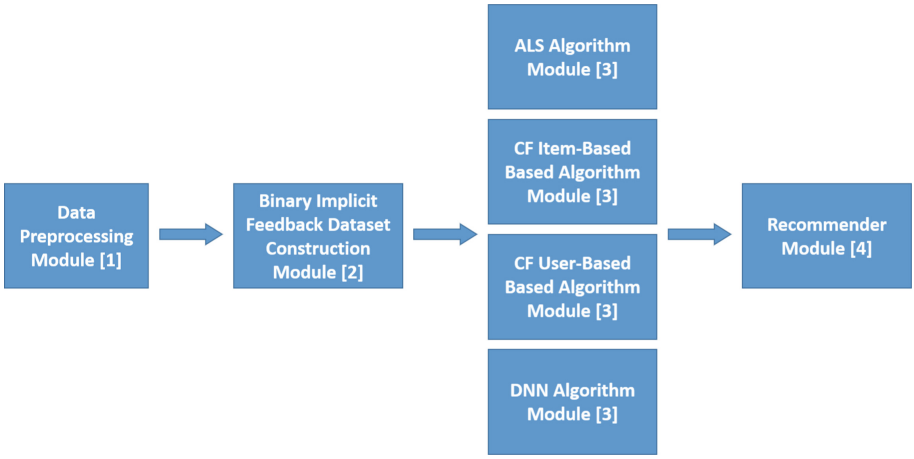

Fig. 2. The proposed business workflow architecture for a book E-commerce website Recommendation Engine _ Modular perspective

This section explains the proposed business workflow methodology followed throughout this study. Figure 1 shows you the data flow extracted from users' interactions in a book website, and Fig. 2 shows you the modular architecture of the software developed in order to produce recommendations; Each number in Fig. 1 refers to the corresponding module in Fig. 2. We are going to explain the business workflow from the modular perspective in detail below:

Data Pre-processing Module: The provided dataset by the book E-commerce website is in raw form. However, for this dataset to be helpful and used as an input to our recommendation algorithms, it should go through the preprocessing stage. the Data Preprocessing module filters out noisy data from the raw dataset. Essentially, we have user interactions from zero to the total number of book items; however, most users have a meager interaction rate with the E-commerce website. For this particular reason, we only want to analyze relatively high interacted users in this study. That is why we picked a threshold in which any user who has interacted (purchased) twenty books or more is counted and considered in our dataset. This module converts the raw data offered by the book E-commerce company into a dataset that can be fed to the Binary Implicit Feedback Dataset Construction Module.

Binary Implicit Feedback Dataset Construction Module: Binary Implicit feedback approach is utilized where explicit feedback information is not accessible or available. Let U and I represent the size of users' sets and items' sets, respectively. We define the user-item interaction matrix $Y \in R^{U \times I}$ from users' binary feedback as:

$$y_{ui} = \begin{cases} 1, \text{Interaction observed;} \\ 0, \qquad \text{Otherwise.} \end{cases} \tag{2}$$

The developer or the company could define the interaction identity; in our case, if User u purchased Item i, then an interaction happened between them; therefore, the value is one.

Table 3. Binary User-Item matrix

Users**Items**	Item_1	Item_2		Item_I
User_1	1	0	...	1
User_2	1	0	...	1
...	...	...	...	...
...	...	...	...	...
User_U	1	0	...	1

As illustrated in Table 3, this would be an output example of this module, where a cell's value is one if the corresponding user did actually purchase the corresponding book item and zero otherwise. And we construct for each user a vector that has a space for all available book items in the book store.

In this context, a zero value does not indicate that $user_u$ dislikes $item_i$; It could mean the user's unawareness of this item. This complicates learning from implicit data because it only gives noisy indications about user preferences. Unobserved entries may be unavailable due to the natural scarcity of negative feedback, while observable entries show users' interest in items.

ALS Algorithm Module: One approach for collaborative filtering which has become popular in recent years is Matrix Factorization. Its potential was first demonstrated in the Netflix Prize competition in 2009. One of the problems that are mainly encountered when using collaborative filtering, contrary to its content-based approaches, is that since they mostly use similarity and neighborhood approaches to recommend items for users, they suffer from a "popularity" problem, which means it primarily recommends popular items since most of the users have given a high rating for the item. Matrix Factorization approaches do not suffer from this problem because they are based on latent factors calculated by the user-item matrix (Fig. 3).

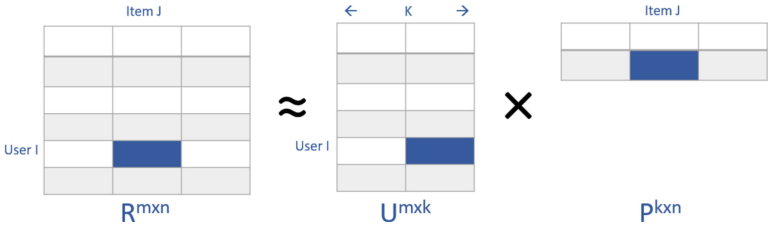

Fig. 3. Matrix factorization [29]

The fundamental concept of Matrix Factorization is illustrated in Fig. 4. The matrix R, which has MxN dimensions, represents our user-item matrix _ where (M, N) denotes the number of (users, items)_ which is decomposed into two MxK and NxK matrices. There are several approaches and implementations of the Matrix Factorization technique for approximating the User-Item relation. Since we are dealing with implicit feedback representing Users-Item interactions using binary values, it is best to use the Alternating Least Square[ALS] approach of matrix factorization [10]. Furthermore, in this study, we are implementing an ALS approach provided in [30] "Collaborative Filtering for Implicit Feedback Datasets."

CF Item-Based Algorithm Module: Despite their apparent problems, classical collaborative filtering approaches are still being used in businesses. Besides being very simple to implement, they are very good at recommending using user interactions.

Since we are using implicit binary feedback for our recommendation system, we also have to choose a good similarity metric to get recommendations. Since Pearson Correlation and Cosine Similarity techniques depend on explicit feedback, we are left with two choices Tanimoto coefficient similarity and Log-Likelihood Similarity, which only account for the presence or absence of the interactions. In our study, we used Log-Likelihood Similarity to calculate users' similarities and limited the recommendations using threshold neighborhood.

CF User-Based Algorithm Module: Item-based and user-based recommenders are both implemented with similar techniques since their fundamental concepts are similarly related to calculating similarity and recommending using any neighborhood method.

Similar to the item-based recommender, we used the Log-Likelihood similarity method for this recommender. Since Log-Likelihood similarity does not consider the ratings given by a user to any items, how does it calculate the similarity between users? To understand this, first, we have to look at Tanimoto coefficient similarity, which is similar to Log Likelihood. Tanimoto coefficient similarity between two users is calculated by dividing the number of items each user has in common, which is their intersection, as shown in Fig. 5, by the sum of the number of items both users have had interaction with, which is their union.

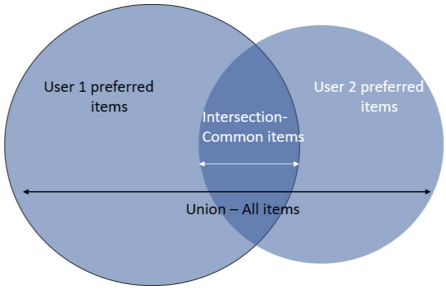

Fig. 4. Tanimoto similarity calculation [29]

Log-Likelihood Similarity calculation is more complex, but its basics are similar to Tanimoto Coefficient Similarity. The only place that they differ is that the Log-Likelihood similarity value shows how unlikely it is that two users' similarity is because of luck.

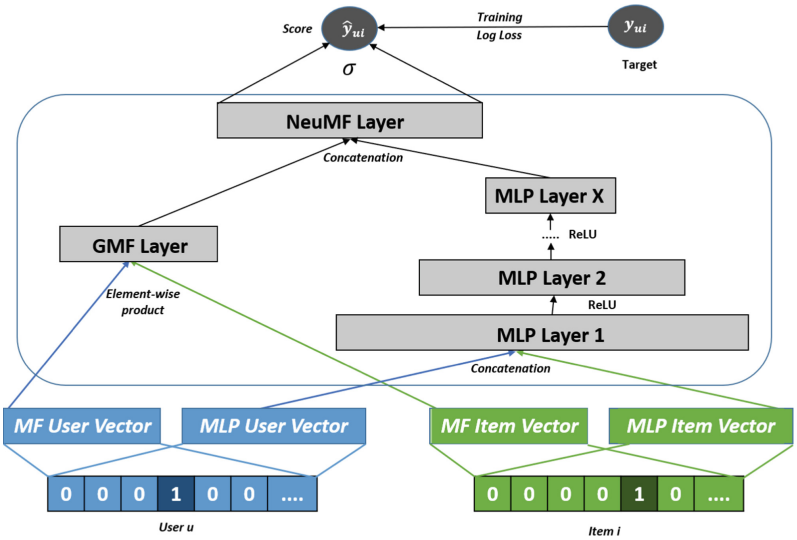

Fig. 5. Deep neural network CF algorithm

Deep Neural Network-based CF Algorithm Module: We are inspired by the deep neural network CF framework that was proposed in the "Neural collaborative filtering" paper published in 2017; Hence we followed the same approach mentioned in the paper [12]. Figure 5 illustrates for us the workflow of the Neural Matrix Factorization algorithm; As it is the product of the fusion of two instantiations of the Neural CF framework _– Generalized MF that models the latent feature interaction linearly and Multi-Layer Perceptron model that

employs the non-linearity nature of neural networks to learn the interaction function from data. GMF and MLP models are integrated by concatenating their last hidden layer after learning separate embedding from the sparse user (item) inputs, which enriches the fused NeuMF model with great flexibility.

The formulation of the DNN CF algorithm clarified in Fig. 5 is given as follows [12]:

$$\begin{gathered} \phi^{GMF} = {p_u}^G \odot {q_i}^G, \\ \phi^{MLP} = a_L({W_L}^T(a_{L-1}(..a_2({W_2}^T \begin{bmatrix} {p_u}^M \\ {q_i}^M \end{bmatrix} + b_2)..)) + b_L), \\ \hat{y}_{ui} = \sigma(h^T) \begin{bmatrix} \phi^{GMF} \\ \phi^{MLP} \end{bmatrix}, \end{gathered} \tag{3}$$

where:

1. $({p_u}^G$ and ${p_u}^M, {q_i}^G$ and ${q_i}^M) \rightarrow$ are (user, item) embeddings for GMF, MLP respectively.
2. $(b_x, a_x$ and $W_x) \rightarrow$ are (bias vector, activation function for the x_{th} layer's perceptron, and the weight matrix) respectively.

This model uses Rectifier (ReLU) as an activation function since it is demonstrated by evidence to be non-saturated, reduces the probability of overfitting, and is well-suited for sparse inputs [12,31]. The model is trained by minimizing the pointwise loss between y_{ui} and $\hat{y}_{ui}$, using a pointwise method termed binary cross-entropy loss, which pays attention to implicit feedback's one-class nature; Formulated by:

$$L = -\sum_{(u,i) \in Y \cup Y^-} (1 - y_{ui}) \log(1 - \hat{y}_{ui}) + y_{ui} \log(\hat{y}_{ui}) \tag{4}$$

Adaptive Moment Estimation (Adam) [12,32] is adapted for optimization purposes in NeuMF, which gradually alters the learning rate for each parameter by executing more minor updates for frequently occurring parameters and more significant updates for infrequently occurring parameters; Thus, the developer does not need to tune the learning rate manually.

The output of this module is the top-N book recommendations items for each individual active user. This top-N recommendation list is generated by sorting the predicted scores associated with each negative instance in ascending order.

Recommender Module: We have the recommendation results for each individual implemented recommendation engine until this point. These recommendations are fed to the Recommender Module to obtain the most relevant ones. As a result, we can generate helpful recommendations for active users, helping them explore book items they most likely will like.

More specifically, in the recommender module, we are merging the results from each recommendation engine based on the hit ratio into one joint list. As we join the results, we assign a weight to each recommendation in the result

list based on the corresponding hit ratio score of the source recommender. The score may come from one recommendation engine or two or more recommendation engines. In that case, we take the average hit ratio from these different recommendation engines. Then we rerank the results based on the final hit ratio score and show them to the users. So this way, we can jointly recommend the result from four different recommendation systems.

4 Prototype and Evaluation

4.1 Prototype

This section explains the technologies that enabled us to complete this novel approach for the recommendation system Business Workflows. First of all, we preprocessed and constructed the binary implicit feedback dataset from the data received from the book E-commerce website using python's NumPy and Pandas libraries, converting our data to the form of (user cookie id, book item id, rating) so we can construct a data stack on which we can run all of our recommendation algorithms modules.

With the help of Apache Mahout, we implement both user-based and Item-based collaborative filtering recommendation algorithms employing the great functionality power provided by Mahout. Between Java and Scala programming languages supported by Mahout, Java programming language was our preferred choice. And for implementing Alternating Least Squares (ALS) algorithm, we used Apache Spark, with Python programming language being our choice among supported languages, including Java, Scala, and Python.

Since we followed the "Neural collaborative filtering" paper's approach in constructing our Deep Neural Network Model, we also used their provided code [12,33]. Their source code was written with python version 2.7 as a programming language and Keras V. 1.0.7 with Theano V. 0.8.0 as the backend. Hence, the source code is outdated, allowing us to update it and make it compatible with the latest stable versions _when conducting this study_ of Python, Keras, and Theano.

4.2 Dataset

As stated earlier, we used a database that contains users' behaviors in a book E-commerce website. Table 4 provides some statistics about the dataset before and after preprocessing. The raw dataset is associated with a high noise rate, which prevents extracting the complete interactions as there are many missing user cookie IDs or book item IDs in the raw database.

Table 4. Statistics of the dataset

Dataset	Interactions	Book-items	Users
Before preprocessing	1807669	197261	1327444
After preprocessing	356320	145444	1804

4.3 Test Design

In this particular study, we employed the $leave_one_o ut$ evaluation methodology _which has been used on a large scale in the literature [34–36]_ to measure each recommendation algorithm's performance and understand how good they are. The overall performance of a recommendation list is decided by $HitRatio$ (HR) and $NormalizedDiscountedCumulativeGain$ (NDCG) [37]. We calculated both measures at 10, 20, 30, 40, and 50 top-N recommendation lists for each user and reported the average score. HR basically detects whether the test item is present on the top-N recommendation list or not. If it is present, NDCG locates the test item position in the list assigning higher scores to hits at the top ranks according to this equation:

$$NDCGscore = \frac{\log(2)}{\log(position + 2)} \tag{5}$$

For example, if the test book-item ID for $user_1$ is 222, and the top-10 recommended books list is [101, 3901, 222, ..., ..., $item_{10}$]; Due to the test item's presence in the recommendation list, the hit value is 1, and therefore the NDCG value would be 0.5.

4.4 Experimental Results

The performance outputs of each recommendation algorithm conducted on our book purchase database are shown in the experimental results. One of this study's purposes was to illustrate how DNN-Based recommendation algorithms perform better than the well-known classical algorithms because of the great flexibility provided by the neural networks allowing us to address several classical recommendation algorithms challenges.

First, we will introduce the classical recommendation algorithms' performance results for HR and NDCG metrics at the top [10, 20, 30, 40, 50] recommendation lists.

Table 5. Classical recommendation algorithm performance

Top-N \Alg.	IBCF	UBCF	ALS
HR @10	0.0055	0.0079	0.0061
HR @20	0.0071	0.0122	0.0112
HR @30	0.0082	0.0159	0.0151
HR @40	0.0094	0.0186	0.0218
HR @50	0.0114	0.0207	0.0250

Top-N \Alg.	IBCF	UBCF	ALS
NDCG @10	0.0035	0.0043	0.0028
NDCG @20	0.0038	0.0055	0.0045
NDCG @30	0.0042	0.0062	0.0054
NDCG @40	0.0043	0.0068	0.0069
NDCG @50	0.0047	0.0071	0.0075

As shown in Table 5, we can observe the low-quality recommendations of the classical recommendation algorithms they produce when they run on top of a highly sparse dataset with binary implicit feedback. The item-based collaborative algorithm was the worst performer. On the other hand, user-based collaborative filtering and Alternating Least squares algorithms compete to show the best results among the classical algorithms, so we notice that for lower values of (N), UBCF is slightly better, but with increasing (N), ALS is showing better results. Now that we reported the average HR and NDCG score performance of the traditional algorithms, we move on to introduce the performance scores of the DNN-Based recommendation algorithm. We set the number of epochs to 20; We report only the performances of the initial model and the best model.

Table 6. DNN-based recommendation algorithm performance

Top-N	Initial HR	Best iteration
HR @10	0.0006	0.0499
HR @20	0.0017	0.0621
HR @30	0.0028	0.0676
HR @40	0.0028	0.0804
HR @50	0.0044	0.0837

Top-N	Initial NDCG	Best iteration
NDCG @10	0.0002	0.0273
NDCG @20	0.0005	0.0302
NDCG @30	0.0007	0.0329
NDCG @40	0.0009	0.0348
NDCG @50	0.0011	0.0409

Table 6 illustrates how well the DNN-Based approach has performed on our sparse dataset. We observe a monotonic increment of the HR and NDCG metrics for the initial model and the best model associated with the increment of N. Consequently, based on what has been shown in the experimental results, we have proven the superiors of the Deep Neural Network approach for generating book recommendations for active users in a book-selling e-commerce website.

Eventually, we create the final recommendation list for the hybrid approach by using the hit ratio as the metric. So, the final result list will favor the results with high ratio hits. Therefore, the hybrid recommender approach will always have highly favored recommended items coming from the available recommendation engines.

5 Conclusion and Future Work

This study designed a novel approach by implementing a hybrid recommendation system for a book e-commerce website by integrating classical collaborative filtering and Deep Neural Network-based methods. Since explicit feedback is often unavailable for items, our data consisted of a preprocessed user-item matrix of binary implicit feedback. Our implemented modules include item-based CF, user-based CF, ALS Matrix Factorization, and Deep Neural Network-based approach. Our item-based and user-based collaborative filtering are implemented in the Mahout framework environment. ALS Matrix factorization is implemented using Apache Spark, and Deep Neural Network is implemented using Keras, Theano, and with the help of the source code from the paper mentioned above. We also evaluated the performances of these recommendation algorithms by calculating the average hit rate and NDCG. We concluded that the Deep Neural Network-based approach had the best result among all other implementations.

In future studies, we will attempt to merge more recommendation algorithms and techniques under the suggested software framework. In addition, multiple tests on several book E-commerce websites will be conducted to investigate the system's performance further.

Acknowledgement. We would like to thank TekhneLogos Company for helping with the dataset and the computational environment.

References

1. Goldberg, K., et al.: EigenTaste: a constant time collaborative filtering algorithm. Inf. Retriev. **4**, 133–151 (2001)
2. Miller, B.N., et al.: PocketLens: toward a personal recommender system. ACM Trans. Office Inf. Syst. **22**(3), 437–476 (2004)
3. Su, X., et al.: A survey of collaborative filtering techniques. Adv. Artif. Intell. **2009**, 2 (2009)
4. Linden, G., et al.: Amazon. com recommendations: item-to-item collaborative filtering. IEEE Internet Comput. **7**(1), 76–80 (2003)
5. Claypool, M., et al.: Combining content-based and collaborative filters in an online newspaper. In: Proceedings of the ACM SIGIR 1999 Workshop on Recommender Systems: Algorithms and Evaluation, Berkeley, California, ACM (1999)
6. Gunes, I., et al.: Shilling attacks against recommender systems: a comprehensive survey. Artif. Intell. Rev. **42**(4), 767–799 (2014)
7. Pandey, A.K., Rajpoot, D.S.: Resolving cold start problem in recommendation system using demographic approach, pp. 213–218 (2016)
8. Papagelis, M., et al.: Qualitative analysis of user-based and item-based prediction algorithms for recommendation agents. Eng. App. Artif. Intell. **18**(7), 781–789 (2005)
9. Bagchi, S.: Performance and quality assessment of similarity measures in collaborative filtering using mahout. Proc. Comput. Sci. **50**, 229–234 (2015)
10. Bokde, D., et al.: Matrix factorization model in collaborative filtering algorithms: a survey. Proc. Comput. Sci. **49**, 136–146 (2015)

11. kumar Bokde, D., et al.: Role of matrix factorization model in collaborative filtering algorithm: a survey. ArXiv abs/1503.07475 (2015)
12. He, X., et al.: Neural collaborative filtering. In: WWW 2017, Republic and Canton of Geneva, CHE, International World Wide Web Conferences Steering Committee, pp. 173–182 (2017)
13. Tran, T., Cohen, R.: Hybrid recommender systems for electronic commerce. In: Proceedings of Knowledge-Based Electronic Markets, Papers from the AAAI Workshop, Technical report WS-00-04, vol. 40. AAAI Press (2000)
14. Burke, R.: Knowledge-based recommender systems. Encycl. Libr. Inf. Syst. **69**, 175–186 (2000)
15. Burke, R.: Hybrid recommender systems: survey and experiments. User Model. User-Adapt. Interact. **12**(4), 331–370 (2002)
16. Smyth, B., Cotter, P.: Personalized tv listings service for the digital tv age. Knowl. Based Syst. **13**, 53–59 (2000)
17. Basu, C., et al.: Recommendation as classification: using social and content-based information in recommendation. In: AAAI/IAAI (1998)
18. Uzun-Per, M., et al.: Scalable recommendation systems based on finding similar items and sequences. Concurr. Comput. Pract. Exp. **2022**, e6841 (2022)
19. Uzun-Per, M., et al.: Big data testing framework for recommendation systems in e-science and e-commerce domains. In: 2021 IEEE International Conference on Big Data (Big Data), pp. 2353–2361. IEEE (2021)
20. Uzun-Per, M., et al.: An approach to recommendation systems using scalable association mining algorithms on big data processing platforms: a case study in airline industry. In: 2021 International Conference on INnovations in Intelligent SysTems and Applications (INISTA), pp. 1–6. IEEE (2021)
21. Tas, K., et al.: On the implicit feedback based data modeling approaches for recommendation systems. In: 2021 International Conference on Electrical, Communication, and Computer Engineering (ICECCE), pp. 1–6. IEEE (2021)
22. Duzen, Z., Aktas, M.S.: An approach to hybrid personalized recommender systems. In: 2016 International Symposium on INnovations in Intelligent SysTems and Applications (INISTA), pp. 1–8. IEEE (2016)
23. Aktas, M.S., et al.: A web based conversational case-based recommender system for ontology aided metadata discovery. In: Fifth IEEE/ACM International Workshop on Grid Computing, pp. 69–75. IEEE (2004)
24. Arpacı, A., Aktaş, M.: Investigation of different approaches for recommendation system. In: ELECO 2018 (2018)
25. Olmezogullari, E., Aktas, M.: Representation of click-stream datasequences for learning user navigational behavior by using embeddings. In: 2020 IEEE International Conference on Big Data (Big Data), pp. 3173–3179. IEEE (2020)
26. Uygun, Y., et al.: On the large-scale graph data processing for user interface testing in big data science projects. In: 2020 IEEE International Conference on Big Data (Big Data), pp. 3173–3179. IEEE (2020)
27. Oz, M., et al.: On the use of generative deep learning approaches for generating hidden test scripts. Int. J. Softw. Eng. Knowl. Eng. IJSEKE. **31**(10), 1447–1468 (2021)
28. Olmezogullari, E., Aktas, M.: Pattern2vec: representation of clickstream data sequences for learning user navigational behavior. Concurr. Comput. Pract. Exp. **34**(9), e6546 (2022)
29. Bokde, D., et al.: Role of matrix factorization model in collaborative filtering algorithm: a survey. arXiv:abs/1503.07475 (2015)

30. Hu, Y., et al.: Collaborative filtering for implicit feedback datasets. In: 2008 Eighth IEEE International Conference on Data Mining, pp. 263–272 (2008)
31. Elkahky, A.M., et al.: A multi-view deep learning approach for cross domain user modeling in recommendation systems. In: Proceedings of the 24th International Conference on World Wide Web. WWW 2015, Republic and Canton of Geneva, CHE, International World Wide Web Conferences Steering Committee, pp. 278–288 (2015)
32. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. CoRR abs/1412.6980 (2015)
33. He, X., Liao, L., Zhang, H., Nie, L., Hu, X., Chua, T.S.: Neural collaborative filtering source code. https://github.com/hexiangnan/neural_collaborative_filtering
34. Bayer, I., et al.: A generic coordinate descent framework for learning from implicit feedback. In: Proceedings of the 26th International Conference on World Wide Web. WWW 2017, Republic and Canton of Geneva, CHE, International World Wide Web Conferences Steering Committee, pp. 1341–1350 (2017)
35. He, X., et al.: Fast matrix factorization for online recommendation with implicit feedback. In: Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval. SIGIR 2016, New York, NY, USA, Association for Computing Machinery, pp. 549–558 (2016)
36. Rendle, S., et al.: BPR: Bayesian personalized ranking from implicit feedback. arXiv preprint arXiv:1205.2618 (2012)
37. He, X., et al.: Trirank: Review-aware explainable recommendation by modeling aspects. In: Proceedings of the 24th ACM International on Conference on Information and Knowledge Management. CIKM 2015, New York, NY, USA, Association for Computing Machinery, pp. 1661–1670 (2015)