

Recommendation System Using Neural Collaborative Filtering and Deep Learning



Vaibhav Shah, Anunay, and Praveen Kumar

Abstract Recommender systems have transformed the nature of the online service experience due to their quick growth and widespread use. In today's world, the recommendation system plays a very vital role. At every point of our life, we use a recommendation system from shopping on Amazon to watching a movie on Netflix. A recommender system bases its predictions, like many machine learning algorithms, on past user behavior. The goal is to specifically forecast user preference for a group of items based on prior usage. The two most well-liked methods for developing recommender systems are collaborative filtering and content-based filtering. Somehow, we were using the traditional methods, named content-based filtering (CB) and collaborative-based filtering (CF), which are lacking behind because of some issues or problems like a cold start and scalability. The approach of this paper is to overcome the problems of CF as well as CB. We built an advanced recommendation system that is built with neural collaborative filtering which uses implicit feedback and finds the accuracy with the help of hit ratio which will be more accurate and efficient than the traditional recommendation system.

Keywords Recommendation system · Neural collaborative filtering · Explicit feedback · Implicit feedback

1 Introduction

Search engines and recommendation systems have become an effective approach to generating relevant information in a short amount of time, thanks to the exponential increase of digital resources from the Internet. A recommendation system is a crucial tool for reducing information overload [1]. Intelligent tools for screening and

V. Shah (✉) · Anunay

Department of Computer Science and Engineering, Parul University, Vadodra, India

e-mail: shahvaibhav348@gmail.com

P. Kumar

Department of Computer Science and Engineering, Indian Institute of Technology, Chennai, India

e-mail: praveen2221cs11@iitp.ac.in

© The Author(s), under exclusive license to Springer Nature Singapore Pte Ltd. 2023
Y. Singh et al. (eds.), *Proceedings of International Conference on Recent Innovations in Computing*, Lecture Notes in Electrical Engineering 1011,
https://doi.org/10.1007/978-981-99-0601-7_10

109

selecting Websites, news articles, TV listings, and other information are among the latest developments in the field of recommendation systems. Such systems' users frequently have a variety of competing needs. There are many variations in people's personal tastes, socioeconomic and educational backgrounds, and personal and professional interests. Therefore, it is desirable to have customized intelligent systems that process, filter, and present information in a way that is appropriate for each user of them. Recommendation System Using Neural Collaborative filtering, Traditionally, relied on clustering, KNN, and matrix factorization techniques.

Deep learning has outstanding success in recent years in a variety of fields, from picture identification to natural language processing. The traditional approach for recommendation systems is content filtering and collaborative filtering. Content filtering is used broadly for creating recommendation systems that use the content of items to create features that contest the user profile. Items are compared to the previous item which is liked by the user, and then, it recommends which is the best match to the user profile [2]. Collaborative-based filtering (CF) is the most popular method for recommendation systems which exploits the data which is gathered from user behavior in the past (likes and dislikes) and then recommends the item to the user.

Collaborative filtering suffers from a cold start, sparsity, and scalability [3]. CF algorithms are often divided into two categories, such as memory-based methods (also known as nearest neighbor's methods) and model-based approaches. Memory-based approaches attempt to forecast a user's choice based on the evaluations of other users or products who have similar preferences. Locality-sensitive hashing, which implements the closest neighbor's method in linear time, is a common memory-based method methodology. On the other hand, modeling methods are developed with the help of data mining and ML techniques to reveal patterns or designs based on a training set [4]. However, an advanced recommendation system uses deep learning as it is more powerful than your traditional methods. Deep learning's ability has also improved recommendation systems. Deep learning capability to grasp nonlinear and nontrivial connections between consumers and items, as well as include extensive data, makes it practically infinite, and consequential in levels of recommendation that many industries have so far achieved. Complex deep learning systems, rather than traditional methods, power today's state-of-the-art recommender systems like Netflix and YouTube.

2 Related Work

2.1 *Explicit Feedback*

In recommendation systems, explicit feedback is in the form of unswerving, qualitative, and measurable responses from users. Amazon, as an illustration, permits customers to rate their purchases on a scale of 1 to 10. These ratings come straight

from the customers, allowing Amazon to quantify their preferences. The thumbs-up button on YouTube is yet another example of explicit feedback from users [5]. However, the problem with this feedback is that it is seldom. Remember when you hit the like button on YouTube or contributed a response (in the form of a rating) to your online purchases? Probably not. The count of videocassettes you specifically rate is lesser than the number of videos you fob watch on YouTube.

2.2 Implicit Feedback

Implicit feedback is collected tortuously through user communications and works as a substitution for user decisions. For example, even if one does not rate the videos explicitly, the videos one watches on YouTube are utilized in the form of implicit feedback to customize recommendations for that user. Let us look at another example of implicit feedback: The products you have window-shopped on Amazon or Myntra are utilized to propose additional items that are similar to them. Implicit feedback is so common that many people believe it is sufficient.

Implicit feedback recommenders allow us to modify recommendations in here and now in short in real time, with every single hit and communication or interaction. Today, implicit feedback is used in online recommender systems, allowing the model to align its recommendations in real time with each user interaction. Though, implicit feedback has its deficiencies as well. Unlike explicit feedback, every interaction is assumed to be positive, and we are unable to capture negative preferences from users. How do we capture negative feedback? One technique that can be applied is negative sampling, which we will go through in a later section.

2.3 Collaborative Filtering

It is a technique of filtering items that a user might enjoy based on the response from other users. The cornerstone of a personalized recommender system is collaborative filtering, which involves modeling users' preferences on products grounded on their prior interaction (ex, ratings, and hits). The collaborative filtering (CF) task with implicit feedback is a common term for the recommendation problem, with the goal of recommending a selection of items to users [6].

Tapestry was one of the first collaborative filtering-based recommender systems to be implemented. The explicit opinions of members from a close-knit community, such as an office workgroup [7], were used in this method. For Usenet news and videos, the GroupLens research system [8, 9] provides a pseudonymous collaborative

filtering approach. Ringo [10] and video recommender [11] are emails and Web-based systems for making music and movie suggestions, respectively.

Transforming Dataset into an Implicit Feedback Data

As the previously stated, we will be using implicit feedback to train a recommender system. The MovieLens dataset, on the other hand, is based on explicit feedback. To achieve this, we will just binarize the ratings to make them ‘1’ (positive class) or ‘0’ (negative class). A value of ‘1’ indicates that the user has engaged with the piece, while a value of ‘0’ indicates that the user has not.

It is crucial to note that providing implicit feedback changes the way our recommender thinks about the problem. Instead of attempting to forecast movie ratings, we are attempting to forecast whether the user will interact with each film, to present users with the films that offer the highest probability of interaction. After finalizing our dataset, we now have a problem because every example dataset falls under the positive class. We would also need negative examples to train our model because we anticipate users are not interested in such films.

3 System Model Approach (NCF)

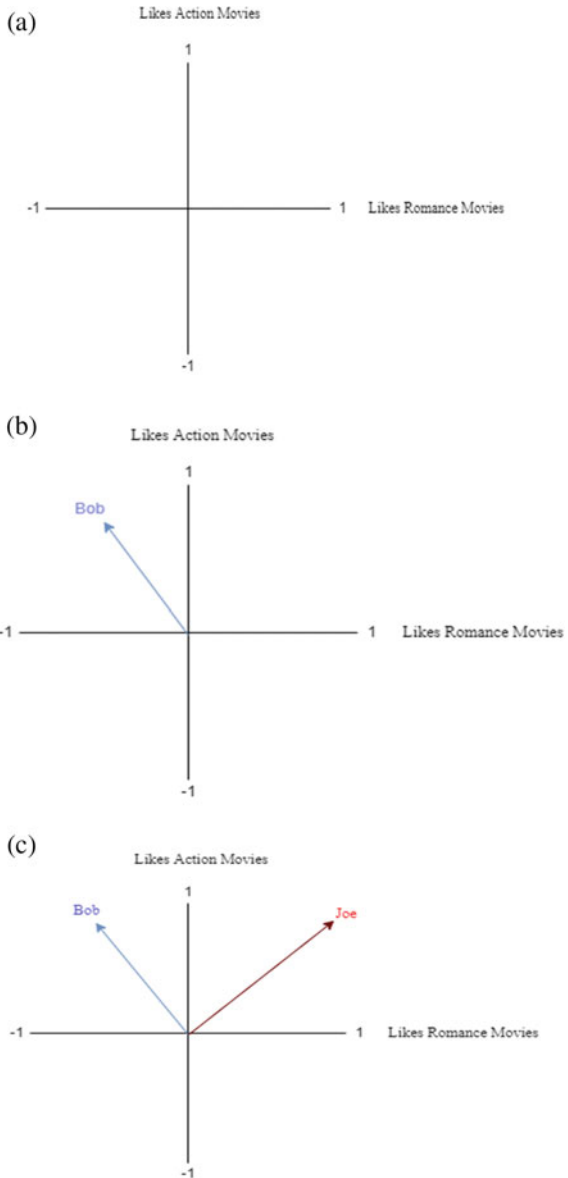
Although there are other deep learning architectures for recommendation systems, we believe that the structure that he-et-al. have proposed is the most manageable to implement and is also the most straightforward one. Most recommendation systems are based on content-based, collaborative-based filtering, or hybrid filtering which are nice models but not as much as they have their disadvantages which let them down somewhere.

To overcome the disadvantages of these models, we come up with a powerful recommendation system using neural collaborative filtering which uses implicit feedback and finds the accuracy with the help of hit ratio which will be more accurate and efficient than the traditional recommendation system. The model we built (recommendation system) gives the best result and accuracy for the search of movies and is similar to it for the recommendation.

4 User Embeddings

Before we get into the model’s architecture, let us get acquainted with the concept of embeddings. The similarity of vectors from a higher-dimensional space is captured by embedding a low-dimensional space. Let us take a closer look at user embeddings to better understand this notion. Let us say, we aim to serve our visitors based on their preferences for two genres of movies: action and fictional films. Assume the first dimension to represent the user’s preference for action films and the second dimension as their preference for fictional films (Fig. 1).

Fig. 1 a User embedding (2-dimensions). **b** Representation of Bob in the embedding Joe will be our next user. Joe enjoys both action and romance films. **c** Representation of Bob and Joe in the embedding Joe, like Bob, is represented by a two-dimension vector above



Assume Bob to be our very first user. He enjoys action films and nonetheless dislikes romance films. We place Bob in a two-dimensional vector according to his preferences.

An embedding is a name for this two-dimensional space. Embedding reduces the size of our users in a way that they could be denoted in a significant way in a

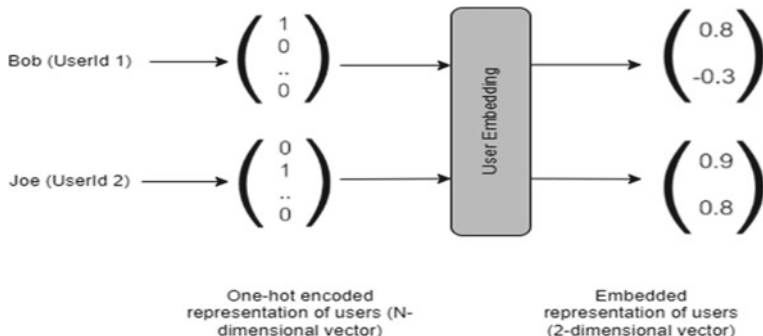


Fig. 2 Representation of users (N-dimension vector) in two-dimensional space

low-dimensional space. Users who have identical movie choices are grouped in this embedding (Fig. 2).

Of course, we are not inadequate to serve our users in only two dimensions. To represent our consumers, we employ any number or a count of dimensions. At the expense of model complexity, a larger amount of dimensions would help us capture the attributes of each user more precisely. We will use eight dimensions in our code.

5 Learned Embeddings

Furthermore, we will define the features of the objects (movies) in a lower-dimensional space using a second object embedding layer. You may wonder how we might get the embedding layer’s weights so that it can present an accurate picture of users and items. We yourself generated embedding in the prior example using Bob and Joe for action and romantic movies. Is it possible to learn such decisions automatically? The answer is collaborative filtering (cf): We can recognize related users and movies by using the rating dataset and generating item and user embeddings based on existing ratings.

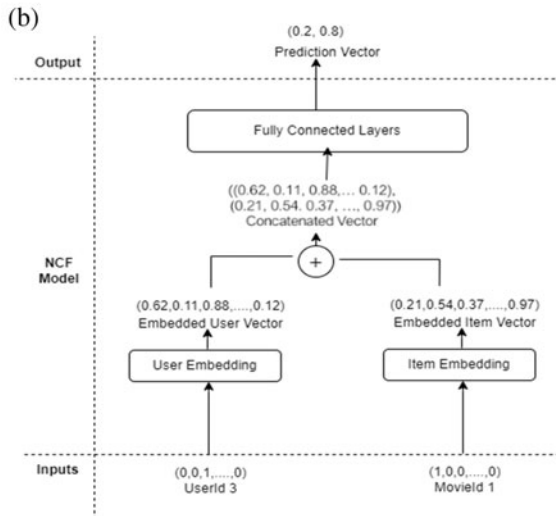
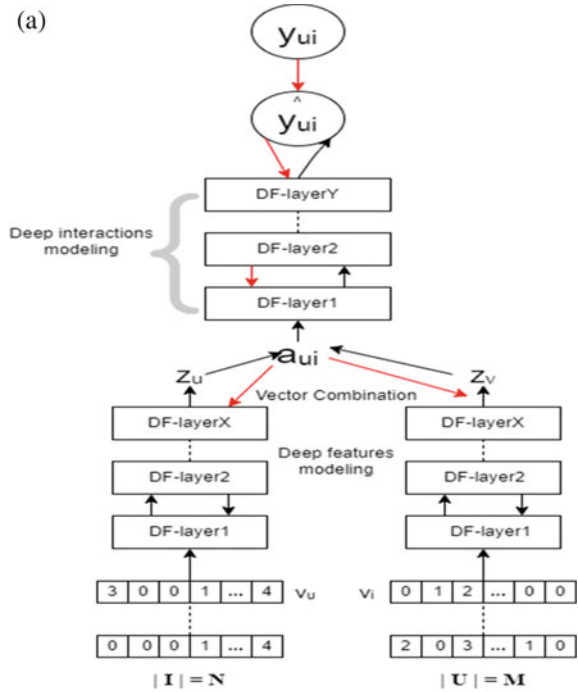
6 Architecture of Model

Now, since we have a good understanding of embedding, we can specify the architecture of a model. As you can see clearly, the item and user embeddings are crucial to our model. Let us have a look at the model architecture using the training sample below:

UserId: 3 movieId: 1 interacted: 1 (Fig. 3).

The embedding layer, which is a fully connected layer that converts the sparse representation into a dense vector, is located above the input layer. Then, the user and

Fig. 3 a Visual representation of the model architecture. **b** Neural collaborative filtering framework



item embeddings are fed to the multi-layered neural network which we call the neural collaborative filtering layers. This layer maps the latent vectors to their prediction scores. The capability of the model depends on the size of the final hidden layer X . In the final output layer, the prediction score \hat{y}_{ui} is present, and the model is trained so as to minimize the loss between y_{ui} and \hat{y}_{ui} . Item and user vectors for $movieId = 1$ and $userId = 3$ are then one-hot encoded as inputs to the model. The true mark (interacted) is 1 because this is a positive sample (the video was genuinely rated by the user).

The item input vector and the user input vector are, respectively, served to item embedding and user embedding, resulting in shorter, denser item, and user vectors. The embedded item and user vectors are amalgamated, here traversing through a sequence of totally connected layers that yield a prediction vector. Finally, we use a sigmoid function to arrive at the best possible class. Because $0.8 > 0.2$, the most likely class is 1 (positive class) in the case above.

7 Evaluating the Model

Our model is currently being trained and is ready to be estimated/evaluated via the test dataset. We evaluate our models in traditional machine learning (ML) projects using measures like accuracy (for classification tasks) and RMSE (for regression tasks) or MAE. For estimating recommender systems, such metrics are far too simplistic. We must first understand how advanced recommender systems are used to define suitable and meaningful metrics for evaluating them.

Take a look at Amazon’s Website, which also has a list of recommendations (given below) (Fig. 4).

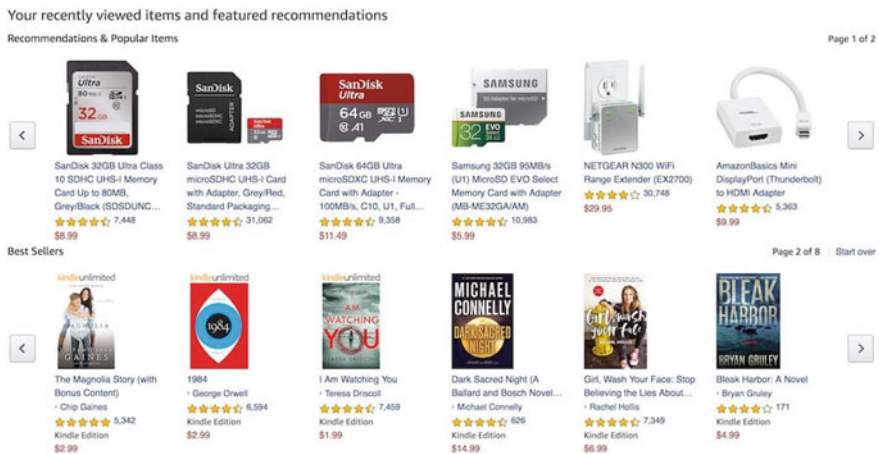


Fig. 4 Snapshot of Amazon’s Website providing recommendations

The idea here is that the user is not required to interact with each and every item on the suggestion list. Rather, we simply want the user to communicate/interact with at minimum a single element from the list of recommendations; if the user does that the recommendations will work perfectly. To replicate this, use the assessment guidelines below to get a list of ten recommended items per user.

- Choose or pick 99 items arbitrarily per user that they have not associated (interact) with up till now.
- Add the over 99 items alongside with the test item (the item that the user actually interacted with). The absolute is presently 100 items.
- Apply algorithm to the above 100 objects and rank them based on their forecast prospects.
- Select top ten items from the above rundown of 100 items of the list. If the test data item is already in the topmost ten, we call it a ‘hit.’
- Repeat the procedure for all users. The average hits are then used to calculate the hit ratio.

This hit ratio is generally used for evaluating recommender systems (Table 1).

$$\frac{\text{\#Of Cache Hits}}{(\text{\#Of Cache Hits} + \text{\#Of Cache Misses})} = \text{Hit Ratio}$$

OR

$$\text{Hit Ratio} = 1 - \text{Miss Ratio}$$

8 Result and Discussion

Alongside the rating, a timestamp column is present that displays the date and time when the review was submitted. Via this timestamp column, we will use the leave-one-out methodology to complete our train test split strategy. The most recent review is used as the test set for each user, while the remaining is used as training data (refer to Fig. 5).

A total of 38,700 movies have been reviewed by people. The user’s most recent film review was for the 2013 blockbuster, Black Panther. For this user, we will utilize this movie as the testing data and the remaining rated movies as training data. When training and grading recommender systems, this train–test split strategy is widely utilized. We could not make a random split.

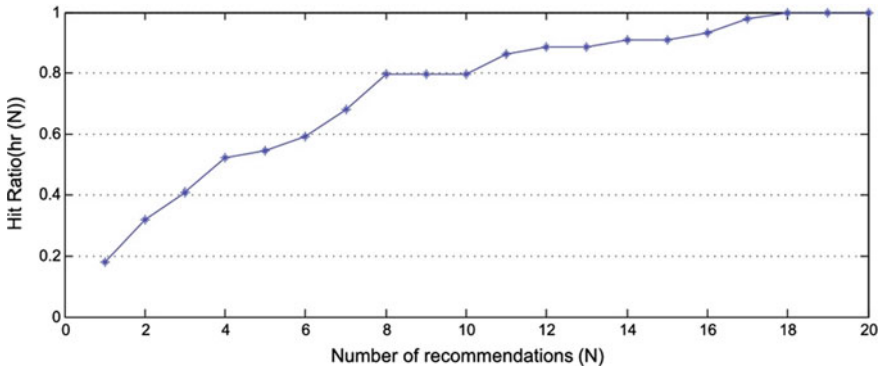
Because we could be using a user’s current review for training and older ones for testing. With a look-ahead bias, it will induce data leakage, and the trained model’s performance will not be generalizable to real-world performance (Graph 1).

Table 1 Number of recommendations versus calculated hit ratio

Number of recommendations (N)	Hit ratio
1	0.19
2	0.30
3	0.40
4	0.56
5	0.58
6	0.60
7	0.63
8	0.80
9	0.80
10	0.80
11	0.89
12	0.90
13	0.90
14	0.95
15	0.95
16	0.97
17	0.99
18	1.00
19	1.00
20	1.00



Fig. 5 Splitting pattern of training and test data



Graph 1 Number of recommendations versus calculated hit ratio

9 Conclusion

Collaborative filtering makes recommendations by simultaneously comparing similarities between people and items, addressing some of the drawbacks of content-based filtering. Serendipitous suggestions are now possible; in other words, collaborative filtering algorithms can suggest an item to user A based on the preferences of a user B who shares those interests. Furthermore, instead of relying on manually designing features, the embeddings can be learned automatically. In this paper, we demonstrated the modeling of a movie recommendation system using the principle of neural collaborative filtering (NCF) with a hit ratio of 86% which is more accurate than the other recommendation models/systems. Our model uses implicit feedback and finds the accuracy with the help of hit ratio which will be more accurate and efficient than the traditional recommendation system. The model complexity has also turned out to be less compared to the other state-of-the-art models.

The model can help users discover new interests. In an isolation, the ML system might not be aware that the user is interested in a certain item, but the model might still suggest it since other users who share that interest might be. Recommendation systems have grown to develop the most indispensable source of authentic and relevant information source in the ever-expanding world of the Internet. The relatively simple models consider a couple or a low range of parameters while the more complex ones make use of a much wider range of parameters to filter results, making them more user-friendly. With the knowledge of advanced techniques like the firefly algorithm and the restricted Boltzmann machine, robust recommendation systems can also be developed. This could be an influential measure toward the further enhancement of the current model to make it more efficient to use, increasing its business value even more (Table 2).

Table 2 Comparison of various state-of-the-art models

S. No.	Model	Model accuracy (%)
1	NCF	86
2	Firefly	84
3	Hybrid	76
4	Collaborative-based filtering	65
5	Content-based filtering	63

References

1. Gao T, Jiang L, Wang X (2020) Recommendation system based on deep learning. In: Barolli L, Hellinckx P, Enokido T (eds) *Advances on broad-band wireless computing, communication and applications*. BWCCA 2019
2. Wei J, He J, Chen K, Zhou Y, Tang Z Collaborative filtering and deep learning based recommendation system for cold start items. <https://doi.org/10.1016/j.eswa.2016>
3. Shah V, Kumar P (2021) Movie recommendation system using cosine similarity and Naive Bayes, s (IJARESM) 9(5). ISSN: 2455–6211
4. Lee H, Lee J (2019) Scalable deep learning-based recommendation systems. Article in *ICT Express*, published June 2019
5. Dooms S, Pessemier TD, Martens L (2011) An online evaluation of explicit feedback mechanisms for recommender systems, WEBIST 2011. In: *Proceedings of the 7th international conference on web information systems and technologies*, Noordwijkerhout, The Netherlands
6. He X, Liao L, Zhang H, Nie L, Hu X, Chua TS (2017) Neural collaborative filtering. In: *Proceedings of the 26th international conference on world wide web (WWW '17)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, pp 173–182
7. Goldberg D, Nichols D, Oki BM, Terry D (1992) Using collaborative filtering to weave an information tapestry. *Communications of the ACM*. December
8. Konstan J, Miller B, Maltz D, Herlocker J, Gordon L, Riedl J (1997) GroupLens: applying collaborative filtering to usenet news
9. Resnick P, Iacovou N, Suchak M, Bergstrom P, Riedl J (1994) GroupLens: an open architecture for collaborative filtering of netnews
10. Shardanand U, Maes P (1995) Social information filtering: algorithms for automating ‘Word of Mouth’
11. Hill W, Stead L, Rosenstein M, Furnas G (1995) Recommending and evaluating choices in a virtual community of use
12. Ma H, King I, Lyu MR (2011) Learning to recommend with explicit and implicit social relations” Facing the cold start problem in recommendation systems. *Expert Syst Appl* 41(4):2065–2073
13. Jeong CS, Ryu KH, Lee JY, Jung KD Deep learning-based tourism recommendation system using social network analysis. Received: 2020.04.02 Accepted: 2020.04.13 Published: 2020.05.31