Industrial Engineering Journal
ISSN: 0970-2555
Volume : 52, Issue 7, No. 4, July : 2023

STUDY ON RECOMMENDATION SYSTEMS: ALGORITHMS AND SCALABILITY STRATEGIES

Aniket Chaudhary, Student, Dept. of Masters of Computer Applications, Vivekanand Education Society's Institute of Technology (VESIT), Mumbai, Maharashtra, India.

Khushi Kansara, Student, Dept. of Masters of Computer Applications, Vivekanand Education Society's Institute of Technology (VESIT), Mumbai, Maharashtra, India.

Dr. Mona Deshmukh, Associate Professor, Dept. of Masters of Computer Applications, Vivekanand Education Society's Institute of Technology (VESIT), Mumbai, Maharashtra, India.

Abstract:
In today's world, the necessity of a recommendation system is growing day by day. Recommendation System is an application of artificial intelligence which is widely utilized on a large scale in sectors like e-commerce, social media, travel, news, healthcare, financial services, etc. There are several models available for recommendation systems. When these models are being deployed on a large scale, it is crucial that the performance is not compromised. In order to overcome this, the models should be scaled and optimized properly. This paper proposes a few algorithms utilized and discusses how to optimize and scale them in order to address the issues encountered while developing recommendation systems for large scale environments.

Keywords: recommendation systems, large-scale environments, scaling, optimization

I. Introduction

Nowadays it is simple to access the vast amount of online information in a quick and comfortable manner because of the recent technological breakthroughs. Users have the ability to provide ratings, reviews, and comments on a wide range of online services and items. However, the problem of online data overload is a result of these advancements [1]. Finding relevant and valuable stuff on the Internet becomes difficult as a result of this data overload. However, the recent development of a variety of techniques with less computational requirements can direct users to the relevant content in a much easier and quicker way [1]. As a result, the recommendation system recently received a lot of attention.

The recommendation system's primary goal is to predict the user's interests and suggest related product items. It is one of the most powerful machine learning systems and it is utilized in a wide range of areas including news, e-commerce, tourism, e-books, movies, music, e-learning, etc [1][5].

As recommendation systems are used on a large scale, issues like delay in response, inaccurate recommendations, scalability, handling of large datasets, ad hoc results, etc can be observed [2][5]. Therefore, it is essential to use appropriate methods or algorithms to build recommendation models and scale them according to the respective needs.

II. Literature Review

A. Paper [1] gives a comprehensive understanding of trends of the recommender system and provides guidelines for its future. It gives brief information on types and problems of recommendation system, algorithmic categorization, applications and datasets used for recommendation systems and research gaps and challenges present in recommendation systems.

B. Paper [3] highlights the significance of optimization in machine learning and its role in training models for achieving desired performance. It discusses various optimization methods, including first-order, second-order, stochastic gradient descent, and metaheuristic algorithms. The paper also discusses convergence properties, computational efficiency, and scalability, and emphasizes specific considerations for applying these methods to machine learning tasks.

C. Paper [4] explains how a distillation loss is used by the teacher and student models to transfer their knowledge to each other. Both models can learn from each other and improve collaboratively using this bidirectional knowledge transfer. Additionally, it uses rank discrepancy-aware sampling techniques adapted to each model's capacities to account for the capacity gap between the teacher and student models.

D. Paper [7] provides information about the collaborative filtering algorithms used for recommender systems. It describes various matrix factorization models which are used to deal with collaborative filtering challenges. It also tells about the procedure used for user based collaborative filtering, item based SVD collaborative filtering and so on along with the understanding of evaluation metrics.

III. Problem Statement

Selecting an appropriate model from a variety of recommendation models which gives accurate results, low latency and low computational cost can be difficult while building a recommendation system. Therefore, the following factors must be taken into account while developing a recommendation model:

***A. Delay in Response*:**

The amount of time it takes for the recommendation model to generate and deliver the users' recommendations is referred to as the delay in response. It is extremely important to take into account since it directly affects user experience and satisfaction.

***B. Inaccurate Results*:**

It implies that the recommended items or suggestions do not align well with the preferences or interests of the user. It means that the recommendations generated by the model are not sufficiently relevant leading to a bad user experience. Insufficient data, cold-start problems, data biases, model limitations, etc are few of the factors that contribute to the inaccurate results [5].

***C. Scalability*:**

It refers to the ability of the system to handle increasing amounts of data, users and items while maintaining the acceptable performance and the quality of recommendation system. Scalability becomes a crucial consideration, when the dataset increases in size as to ensure that the recommendation system remains efficient, responsive and capable of delivering accurate and timely recommendations [4].

***D. Computational Cost*:**

It refers to the amount of computational resources such as processing power, memory and time required to train the model, generate recommendations and perform other necessary operations. Computational cost is an important factor while designing and implementing the recommendation system as it affects system's efficiency and overall performance [5].

This study aims to assist in the optimization of the appropriate model and various scaling strategies to address the mentioned problems that are beneficial in large scale environments.

IV. Research Methodology

There are numerous recommendation system models widely used across all domains and applications. The choice of recommendation system model depends on various factors such as the characteristics of the data, the specific goals of the system, and available computational resources. Some of the methods used to build recommendation models are listed below:

A. Cosine Similarity:

Cosine Similarity is similar to a content-based filtering approach used for recommendation systems. It is one of the most popular techniques used for recommendation system. "Content" refers to a certain item's attributes. These attributes allow us to classify whether two items are similar or not.

For instance, the attributes in a movie recommendation system could be the genre, cast, director names, the description, etc [2]. If the attributes of two movies match or have a high similarity between them, then those movies can be classified as similar movies. This simply implies that if userA enjoys movie1, he or she might also enjoy movie2, which has a high similarity.

Mathematically, cosine similarity is calculated as the dot product of two vectors divided by the magnitude of each vector. If the cosine similarity between two items is 1 or close to 1, then the items are considered to have high similarity between them [2]. Consider two items P and Q, the cosine similarity between these two items are calculated by using the formula given below,

$$\text{Cosine Similarity (P,Q)} = cos\theta = \frac{P.Q}{||P||.||Q||}$$

***B. Matrix Factorization*:**

Matrix factorization is one of the most popular model-based collaborative filtering method. Matrix factorization shows the mapping of both the items and users by using the feature vectors generated from the user's rating patterns for the items [2]. The joint latent features space can be generated by multiplying two different kinds of entities. In a recommendation system, it is used to identify the relationship between the items and user's entities. The main purpose of matrix factorization is to approximate the rating matrix with a low-rank matrix [2]. Low-rank means to decompose the original user-item rating matrix into a smaller dimension matrix. This rating matrix is often sparse, which means that most of the entries are not present due to unavailability of rating from users for some items. So, it becomes essential to handle this type of non-rating pattern [2]. Suppose, the joint feature space is of dimensionality *k*, and user *u* is associated with vector space m and item *i* is associated with vector space n then,

$$u_i = m_u * n_i^T$$

***C. LightGCN*:**

Graph Convolution Network (GCN), a neural network architecture designed for graphs structured data like user-items [8]. GCN is based on the concept of convolutional neural networks (CNNs). The basic idea behind GCN is to aggregate and propagate the local and global context from adjacent(neighboring) nodes like in CNN, convolution layers aggregate local information by applying filtering to local receptive fields and pooling for location invariant [8].

GCN layers typically operate by aggregating the node features from neighboring nodes, applying linear transformations, and applying non-linear activation functions to produce the node embeddings.

LightGCN is a streamlined version of GCN [8]. While GCN considers both direct connections and higher-order connections in the graph, LightGCN simplifies the aggregation by only considering the normalized sum of neighbor embeddings direct connections. This reduces the complexity and makes LightGCN more suitable for large-scale recommendation systems. Many companies like Uber, Airbnb, Pinterest etc. follow a GCN based model for recommendation systems [8].

We have discussed some of the content-based, collaborative based and graph collaborative based recommendation systems above [2][5][8]. They have shown better results over years. But when we talk about usage of the above model in large scale environments or to use the model for edge devices, it is necessary to scale and optimize the models by making adjustments in model configuration and doing hyperparameter tuning along with feature engineering etc accordingly. As we scale models, it is necessary to optimize in order to improve the performance, efficiency, and quality of the recommendation algorithms. There are various methods available in order to make models respond better with less complexity when doing inference in any environment.

***A. Parallelising*:**

Parallelization allows multiple computations to use multiple resources and reduces the overall execution time. Some of the techniques used for parallelising recommendation models are listed below:

1. ***Model Parallelism:*** In this technique, recommendation models are split into parts and processes them separately on different computing units. It's beneficial for complex models or when different parts need separate computations.
2. ***Data Parallelism*:** In this technique, the dataset is divided into multiple subsets, and each subset is processed independently by separate computing units. This approach is suitable when the recommendation model can be trained or evaluated on different parts of the dataset in parallel.It can be used for matrix factorization where the user-item matrix is divided into smaller submatrices and perform factorization on each submatrix concurrently [2].

B. Caching and precomputation:

Caching can be improved by storing the frequently accessed data in a cache. In recommendation, there are few methods, where we can use the caching method. For example, rather than computing item-item similarity again for each request, we can precompute and store the similarity score in a cache [5]. This will make quick retrieval of similarity scores during recommendation generation. We can also precompute user embeddings, item embeddings or other relevant feature**.**

C. Partitioning:

Partitioning refers to the process of dividing the data and computations involved in recommendation tasks across multiple computing resources or nodes [4]. It is a technique used to distribute the workload and enable parallel processing, improving the scalability and efficiency of the system. There are two main types of partitioning in recommendation systems:

1. ***Data Partitioning****:* Data partitioning involves dividing the user-item interaction data or other relevant datasets into subsets that can be processed independently. Common approaches to data partitioning include user-based partitioning, item-based partitioning, range-based partitioning and hash-based partitioning.
2. ***Computation Partitioning:*** Computation partitioning involves dividing the recommendation system computations into separate tasks or components that can be executed independently. This allows different computing units to perform different parts of the computation in parallel.

D. Model Compression:

Large model with numerous learning parameters has better recommendation performance but it requires large computational costs and high inference latency which becomes a major obstacle in real-time. To overcome this issue, different model compression techniques have been used, one of which is knowledge distillation [6].

Knowledge Distillation*:* Knowledge Distillation is a strategy which generates a small student model by transferring knowledge from large teacher models [3]. Different forms of knowledge can be categorized as:

1. ***Response Based Knowledge*:** Student model will learn from the Output layer of the teacher model.
2. ***Feature Based Knowledge****:* Student model will learn from the latent (Hidden) layers of the teacher model.

In the recommendation system, large and accurate teacher models are trained using user-item data. Output of this training generates soft labels instead of hard labels. Soft labels are probability distributions over the items, which is obtained by applying temperature value (T>0) to logits [3].

Student model can become more confident in approximating the probabilities generated by the teacher model by adding temperature value. In the knowledge distillation process, we can balance between exploration and exploitation by just adjusting the temperature.

As we compress or distill the size of the model, the model becomes less complex and there could be change or drop in the accuracy of the model. To evaluate how well our model works, we need to use a proper evaluation metric. Evaluation metric proves a quantitative measure of how well the model is performing by assessing the model's performance against specific metrics. It also helps in fine-tuning the hyperparameters. We have to make sure that after each step or for every subsets, the recommendation prediction should not be impacted. In order to evaluate the prediction accuracy and effective implementation of the various recommendation models, different types of evaluation metrics are used such as:

A. Mean Average Precision at K (MAP@K):

Precision is commonly used to evaluate the accuracy of the generated recommendations [1]. Precision is typically computed by,

$$Precision = \frac{No.\ of\ relevant\ items\ in\ recommendation\ list}{Total\ no.\ of\ recommended\ items}$$

MAP@K is a popular evaluation metric used to assess the quality of ranked recommendation lists [1]. It takes into account both precision and the order of recommendations, providing a more comprehensive measure of recommendation performance. It calculates the average precision at each position up to K in the recommendation list and then takes the mean of these values. A MAP@K of 1 means that all recommended items at each position up to K are relevant, while a value of 0 indicates that none of the recommended items are relevant [1].

B. Mean Average Recall at K (MAR@K):

Recall is commonly used to measure the effectiveness of the system in retrieving relevant items [1]. Recall is typically computed by,

$$Recall = \frac{No.\ of\ relevant\ items\ in\ recommendation\ list}{Total\ no.\ of\ relevant\ items}$$

MAR@K is a metric used to evaluate the average recall of relevant items among the top-K rank of the recommendation list [1]. It is calculated by taking the average of recall values at each position up to K. MAR@K ranges from 0 to 1 similar to MAP@K where 1 means that all relevant items are captured within the top-K positions for each user, while a value of 0 indicates that none of the relevant items are included [1].

C. KL Divergence:

Kullback-Leibler(KL) divergence is used to measure the difference between the output of student model and teacher model [3]. The KL divergence states how much of information is lost when one probability approximates another.

$$KL\ (T\ ||\ S) = \Sigma\ T(x)\ *\ log\ \frac{T(x)}{S(x)}$$

where T & S is output distribution of Teacher model and student model respectively. KL divergence sometimes is combined with other losses like in the training process in order to ensure that the student model not only mimics the teacher model but also makes accurate predictions [3].

D. Mean Squared Error (MSE):

MSE is an evaluation metrics which measures predicted ratings accuracy or scores compared to the actual ratings provided by users [5]. These metrics are particularly relevant when dealing with rating-based recommendation systems.

Average of the squared differences between predicted ratings and actual ratings is calculated using MSE whereas RMSE is the square root of MSE, providing a measure of the average magnitude of the prediction errors in the original rating scale. Better accuracy is indicated by lower MSE and RMSE values, as they reflect smaller differences between predicted and actual ratings [5].

It is important to monitor the evaluation metric. Low score in evaluation can make the model to not align well with the user preferences or provide relevant and accurate recommendations. Low score models also will be dependent on additional resources, which will eventually increase the computational cost and can impact the business.

V. Analysis And Findings

After conducting a survey, we observed that approximately 40% of the people experience a delay in getting recommended products. Moreover, it is also observed that around 55% of the people encounter inaccurate recommendation results. 80% to 90% people find the recommendations useful in order to purchase their desired products. The detailed insights about the survey is shown in the table below:

TABLE I. SURVEY TABLE

Sr No.	Questions	Responses	
		Yes	**No**
1.	Have you ever noticed a delay in getting recommended products on any e-commerce platform?	37.5%	62.5%
2.	Do you believe that any ecommerce websites help you find your desired products through recommendation?	80%	20%
3.	Have you ever encountered recommendations for women's clothing while searching for men's clothing on the website or vice-versa, even though you did not explicitly mention the gender?	55.6%	44.4%
4.	After purchasing a product, do you find the recommended combinations along with the purchased product useful? Example: When you purchase a mobile phone, it recommends you to purchase a combination of screen guard and mobile cover too!	72%	28%
5.	Do you find the Compare functionality of ecommerce websites useful?	90%	10%

Following is the result of knowledge distillation applied on LightGCN model:

A. We used MovieLens 100K [10] dataset having features such as UserID, ItemID, ratings.

B. Timestamp values are not sequential or we can say they are independent, so we drop the column and remain with only UserID, ItemID and rating.
C. For the model we used LightGCN.
D. We evaluated the model with top 10 K and also applied knowledge distillation to compress the model size.
E. Model was trained with an embedding size of 64, learning rate 5e-3 and all the results are the average of 30 iterations.

TABLE II. MODEL EXPERMINENT AND EVALUATION

Model	**Evaluation Metric**	
	recall@10	ndcg@10
LightGCN	0.2080	0.3408

We got recall@10 of 0.2080, which suggests that approximately 20.8% of the relevant items were included in the top 10 recommendations provided by the model. And ndcg@10 of 0.3408 indicates that the ranking of recommended items by the LightGCN model performs significantly better than a random ranking, with 39% of the cumulative gain achieved by the model.

VI. Limitations

Recommendation systems typically rely on user data, including browsing history, purchase behavior, and personal preferences. Collecting and analyzing such data can raise privacy risk, as users may be cautious to share their personal information.
In many cases, recommendation systems face data sparsity, where the available user-item data is limited, leading to sparse matrices or incomplete information. This can make it difficult to derive meaningful patterns and accurate recommendations. Sparse data can result in limited coverage and lower recommendation quality.
Recommendation algorithms tend to favor popular items due to the availability of more user interactions and feedback. This creates bias towards popular items and results in a lack of diversity in recommendations. Less popular items may not receive sufficient exposure, hindering unforeseen discovery for users.

VII. Future Scope

With increasing volume and variety of data, scaling recommendation systems to effectively handle big data becomes crucial. Future advancements may involve leveraging distributed computing frameworks or cloud-based solutions to efficiently process and analyze large-scale data.
Scaling recommendation systems to provide real-time or near real-time recommendations is a significant area of development. By reducing latency and enabling faster processing, recommendation systems can deliver personalized recommendations, enhancing user experiences and responsiveness.
Future scaling efforts may involve developing privacy-preserving recommendation techniques that provide personalized recommendations while ensuring user privacy and data security.

VIII. Conclusion

Recommendation Systems have emerged as powerful tools that enhance user experiences, increase engagement and drive business success across various sectors. In this paper, we have identified the challenges which are faced while deploying recommendation models in large scale environments and have enlightened the scaling techniques which can be implemented to overcome these challenges. Apart from

this, a detailed study is done on some of the popular algorithms used in recommendation systems and the importance of evaluation metric is also highlighted. Overall, this paper provides a comprehensive study of the algorithms used in recommendation systems along with the scaling techniques which can be implemented while deploying recommendation models in large scale environments.

References

[1] Deepjyoti Roy & Mala Dutta (2022), "A systematic review and research perspective on recommender systems", 2196-1115.
[2] Dheeraj kumar Bokde, Sheetal Girase & Debajyoti Mukhopadhyay (2014), "Role of Matrix Factorization Model in Collaborative Filtering Algorithm: A Survey".
[3] Geoffrey Hinton, Oriol Vinyals & Jeff Dean (2015), "Distilling the Knowledge in a Neural Network".
[4] Joeran Beel, Bela Gipp, Stefan Langer & Corinna Breitinger (2016), "Research-paper recommender systems: a literature survey", 1432-1300.
[5] M. Gupta, A. Thakkar, Aashish, V. Gupta and D. P. S. Rathore (2020), "Movie Recommender System Using Collaborative Filtering," International Conference on Electronics and Sustainable Communication Systems (ICESC), Coimbatore, India, pp. 415-420, doi: 10.1109/ICESC48915.2020.9155879.
[6] SeongKu Kang, Dongha Lee, Wonbin Kweon, Hwanjo Yu (2021), "Personalized Knowledge Distillation for Recommender System", Knowledge-Based Systems.
[7] Wonbin Kweon, SeongKu Kang & Hwanjo Yu (2021), "Bidirectional Distillation for Top-*K* Recommender System".
[8] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang & Meng Wang (2020), "LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation".
[9] Y. Koren, R. Bell and C. Volinsky (2009), "Matrix Factorization Techniques for Recommender Systems," in Computer, vol. 42, no. 8, pp. 30-37, doi: 10.1109/MC.2009.263.
[10] https://grouplens.org/datasets/movielens/