Recommender Systems

Onur Porsuk[1], Ayse Basar[2] and Suzan Uskudarli[3]

Abstract— **Recommendation Systems (RS) are increasingly important in today's digital age, as they help to personalize the online experience for users by providing them with tailored content and product recommendations. The users of an application interact with items which may be movies, songs, documents or games. As the data which is produced by users grows, today, a lot of items exist in applications to provide well utilization to users. A recommendation can be either personal or non-personal and depending on this, different methods are implemented to achieve the desired goal. In this research study, an overview of RS domain will be discussed by mentioning the RS types in detail, some methods, dataset types, model building, evaluation, challenges and solutions. Overall, the study aims to provide a comprehensive understanding of the fundamentals of RS, some difficulties and opportunities that lie ahead in this rapidly evolving field.**

I. INTRODUCTION

A recommender is a sophisticated system that tries to recommend items to users in an application. It may be a software tool or technique which provides suggestions for items that are most likely of interest to a particular user [1]. For example, Amazon is one of the largest online shopping companies which uses RS at its core. Amazon provides millions of products to its customers and the developers improve the application day by day to make it capable of serving well for user satisfaction.

There are five types of RS in general: Collaborative Filtering, Content-based Filtering, hybrid approaches, Knowledge-based and Demographic RS. Each method has its own benefits and pitfalls. An individual who works in RS field should be careful in choosing the method for her/his problem. The success of the recommender highly depends on the problem, dataset and objectives. Hence, it is important to make the right decision during development. In an application, there might be millions of users and items and one of the major problems in RS is that there is no real scenario that all users interact with all items. In large applications, users observe only a very small portion of items in the item space. [2] shows that there is high sparsity in some datasets. When the application collects feedback from users after observations, for instance, the most popular way is the 5-star rating, then there will be very little information about the feedback. There are different methods to overcome the missing information in RS, which will be discussed later.

This study aims to do comprehensive research about RS field by adding self-critical thinking and analysis. The sections in the paper include the followings: the next section contains different RS types with their definitions, pros and cons. In Section III, one of the widely used algorithms in RS will be presented. Section IV will be about the feedback types in RS and some challenges that they arouse. Section V will provide another widely used method, latent factor models. The following section includes some additional topics such as negative sampling. Section VII covers some loss types in RS. The next is about the evaluation metrics of a recommender model. The last three sections are some advanced topics in RS field.

II. TYPES OF RECOMMENDATION SYSTEMS

In this section, there will be brief information about each type of RS.

[1]Onur Porsuk, CMPE, Boğaziçi University
[2]Ayse Basar, CMPE, Boğaziçi University
[3]Suzan Uskudarli, CMPE, Boğaziçi University

A. Collaborative Filtering

Collaborative Filtering (CF) is one of the commonly used methods which considers the similarity between either users or items. It consists of two main types, which are memory-based and model-based methods. [3]

In memory-based, the implementation can be either user-based or item-based. In the user-based version, the system tries to calculate how similar are users to each other by considering their past observations. To give an example, in a song stream application, two users A and B may have listened to various songs. If we want to know, whether user A would like a specific song, we can check if user B or other similar users to A have listened to the song or not. If A's top-n similar users listened to the song, we may recommend it to user A. This is just a basic example of user-based CF. For the same user A, if the application wants to decide whether it should recommend an item K to user A, it looks for similar items to K. If user A observed similar items to K, then the system can decide that A will probably observe K too, which is item-based CF. The concept of these techniques is similar, the only difference is the objects. Usually, in real-life scenarios, item-based performs better than user-based CF. [4] presents the utilization of item-based CF in e-commerce industry.

Instead of observation or no observation (1 or 0), a rating value could be used. The calculation is done by the ratings' values instead of interaction information. To measure similarity between users or items, *Cosine* or *Pearson* similarity techniques are options to implement. Study [5] explains that there is a different use case for these two techniques: Cosine similarity ensures that the similarity is not fully decided only by the subset of items (potentially small) the two users have in common. This is not natively present in Pearson but it can be added by significance weighting. Thus, in contexts where users tend to have very different sets of items in their profiles, in principle, Cosine similarity may perform better.

The data that is used is a matrix which contains users and items in rows and columns. This matrix is called *User Rating Matrix (URM)*, where there are rating values in cells. In memory-based type, it is easy to implement and understand to explain results but due to the large matrix, it may not be scalable for performance issues [6]. The second approach is a model-based method in CF. In this approach, instead of measuring the similarity of users or items, Machine Learning (ML) comes in to build a model which would learn the patterns in the dataset. A general diagram of the process was shown in Figure 1.

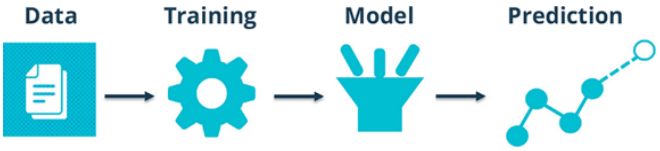

Fig. 1. Model-based CF [7]

In an ML model development process, after training, the model is created and it is utilized to make future predictions. For instance, K Nearest Neighbour (K-NN) or multi-layered neural networks are implemented. Model-based CF works well since it uses the power of ML. In this approach, the URM is not directly brought into the main memory. Instead, ML takes place in learning the similarities between uses and items. Hence, unlike memory-based, method-based consumes less memory. Another advantage of the model-based approach is that it handles the *cold start* problem which is defined as the new user or item problem in the system. When there is a new user which comes into the system, there is no past information and this means the CF model is unable to make a personalized recommendation. Model-based overcomes the cold start problem for new items. Also, with the help of dimensionality reduction techniques, the method can deal with sparsity. As a disadvantage, unlike memory-based, explainability may be difficult due to hidden layers and factors in model-based. It is important to point out that ML algorithms require dense (complete) input to work. Generally, CF's main advantage is that there is no domain knowledge requirement and it can be applied to any domain without additional effort to make the system work.

In my opinion, CF is a powerful solution to implement a recommender for the target domain. I think it is relatively easier to build and as it does not require domain knowledge, it would take a shorter time to value a foundation. Apart from the mentioned reasons, I would personally choose CF when there are observable similarities between users-items and a complex approach is not required. In the practical session of the course, I experienced both item and user-based CF using a movie ratings dataset. My result showed that both techniques predicted *relevant* items to the tested users. Since my dataset was not that large and did not contain so many user-item interactions, the results of these two methods were close to each other.

B. Content-based RS

The content information of items is utilized in this type of RS. Content refers to the characteristic features of items. For example, in a document reading application, there are various attributes of a single document such as topic, author, release date, special tags etc. The advantage is that there is no need to have information about other users' interactions since the content information is used for predictions but domain knowledge is highly required because there should be intense information.

The advantage of Content-based RS over CF is that generally there are both new user and new item cold start problems in CF when using user-based or item-based approaches. However, Content-based copes with only a new user problem, since a new item already provides the necessary information. Depending on the problem and dataset, usually, Content-based RS is more accurate. [8] states that Content-based RSs are more effective in some real-world application domains than their pure CF-based counterparts.

C. Hybrid Approach

In the implementation of a recommender system, hybrid approaches are powerful. Instead of one single method, engineers can use more than one method at the same time. For example, Collaborative Filtering and Content-based RS may be used as a hybrid model, which obtains the strengths of each method and overcomes some disadvantages. In real-world scenarios, usually, a hybrid approach is implemented to build a recommender and it performs better than a single method.

D. Knowledge-based RS

A Knowledge-based RS is based on explicit knowledge about the item assortment, user preferences, and recommendation criteria [9]. This type of RS is applied when CF or Content-based RS is not applicable in specific scenarios. It inherently handles the cold start problem but requires deep information. These RS systems are well suited to complex domains where items are not purchased very often. For example apartments, land or cars. In such scenarios, there are fewer observations than in the other domains.

E. Demographic RS

Demographic RS makes use of the demographic attributes of users to recommend items. It does not require users' ratings or information of items [10]. Hence, it can overcome the cold start problem as Knowledge-based RS. This type of RS can be easily implemented and utilized as CF.

From my point of view, each RS type has different capabilities and if we focus on bringing their skills together to enhance the power of the solution, we will go way further than one single approach. I think that adding deep knowledge information about the domain that we are working on into CF and Content-based RS will have a good impact on recommendations.

III. K-Nearest Neighbour Algorithm

K-Nearest Neighbour (K-NN) is one of the most frequently used algorithms in similarity-based tasks. It has a straightforward way which is easy to understand. Also, it works fast and can be quickly implemented. It basically works on distance measuring of samples (points) in the space. The general steps were shown in Figure 2.

First, the dataset is captured and prepared for training. K-NN requires the K value to work. This is the number of sample observations in each iteration. For example, if K is taken 5, in each iteration, the 5 closest neighbours are

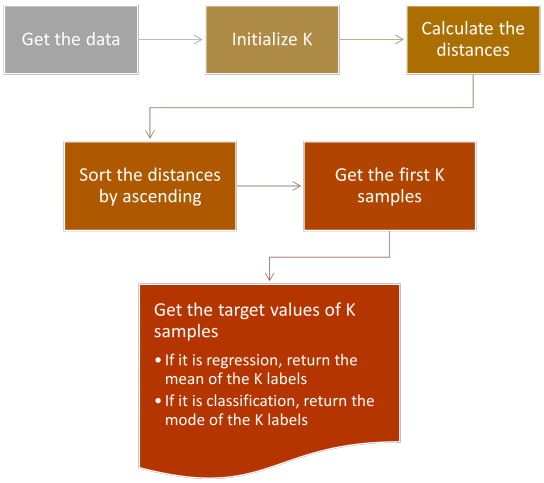

Fig. 2. K-NN - Steps

considered to decide which class the sample belongs to. In distance measurement, different techniques such as *Euclidean, Manhattan* or *Mahalanobis* exist. After deciding K, for each sample in the set, their corresponding closest neighbours are found by measuring the distance. If there is no improvement in the implementation, the naive way is to calculate distances from the observed sample to all other samples in the space. Next, they are sorted by distance and the closest K ones are gathered. Depending on the problem, *classification* or *regression*, the decision of assigning the observed value is done by considering the K closest neighbours. If it is classification, *majority voting* is used, which means the most frequent class is the class of the observed sample. For classification tasks, because classification is based directly on the training examples, the algorithm is also called *Example-based Classification* or *Case-based Classification* [11]. If it is regression, either *mean* or *median* value is decided to be the value of the observed sample.

In the practical session, I experienced K-NN algorithm using a dataset about diamonds. Although the dataset was not very large, I have seen that K-NN usually worked quite fast. During the improvement of my model, I also focused on outlier removal and sampling the dataset to avoid the imbalance problem not to face *bias* in predictions.

IV. Types of Feedback

There are mainly two types of feedback: explicit and implicit. Explicit means that users directly mention their opinions about the item by either giving a rating or leaving a text comment. On the other hand, implicit feedback is the system-monitoring-based information which the application collects. For instance, on a website, implicit feedback can be the time that the user spends on specific items, the clicks, the positions of clicks, the user's preferences, settings etc.

Explicit feedback is more accurate than implicit because it is direct and exact information about the likes or dislikes of the user for the item. But in real-world scenarios, it is challenging to get feedback from users. Unlike explicit, there is abundant information in implicit feedback, and numerous records and log information are produced by the system as it monitors the users' behaviours continually. However, the interpretation of implicit feedback is not as easy as explicit. Because an engineer cannot be sure whether the information indicates that a user liked an item or disliked it. It is challenging to make a structured implicit dataset to use as easily as an explicit one. Apart from these two feedback types, people may use a combination of them in developing recommenders. Usually, it strengthens the feedback's positive impact on performance.

In addition to these three types, there are two additional feedback types which are contextual and social feedback. Context is the environment where users are interacting with the system and contextual feedback includes information such as the time of the day, application version or user location. Social feedback means the use of information about the preferences and actions of a user's social network to make recommendations, e.g. the items that a user's friends have liked or purchased. In study [12], it is mentioned that social networks can provide useful information to understand users' preferences as suggested by social theories such as homophily and influence.

V. Latent Factor Models

URM is highly sparse due to the missing values and many non-observations. Instead of processing the whole matrix, it can be written as sub-matrices multiplication. This is useful to reduce the dimensionality and process smaller sizes of matrices. This approach is called matrix factorization. The models created by this approach are named latent factor models and it reduces the computational time, however, generates some errors because of the transformation. Matrix factorization essentially maps the features into a smaller space and represents each user and item on the space. The item which any user observes is close to the users in the space, i.e. the distance between them is short. The estimated rating formula is shown in Equation 1. In the formula, *u* and *i* represent user and item indices respectively. An estimated rating value equals the multiplication of user and item *latent vectors*.

$$r_{ui} = q_i^T p_u \tag{1}$$

In matrix factorization, there are different methods and Singular Value Decomposition (SVD) is one of the commonly used approaches. The difference between base matrix factorization and SVD is that it depends on what loss function and what properties are desired from the result. But essentially they work in the same way. The formula of SVD is shown in Equation 2. There is the third component in the formula which is a diagonal matrix that contains singular values.

$$A = U\Sigma V^T \tag{2}$$

It is important to note that SVD requires a dense matrix to work. Therefore, engineers must deal with the missing values in URM. Moreover, matrix factorization models learned by SVD have shown to be very prone to *overfitting* [13], which is a common problem in ML tasks. Some *regularization* must be applied to reduce the negative impact of intensive learning.

SVD was the second algorithm which I implemented in the practical session. I think it works well when especially the missing values are handled properly. Also, it had a short execution time which is valuable for some specific domains. In addition to the imputation of missing values, I worked on Pearson and Cosine similarities. I have learnt that it is always critical to have a reason for choosing a method rather than others in projects, which changed my view to think more correctly.

VI. More Topics About Feedback Types

In explicit feedback type, there is a clear structure and it is straightforward to use. However, implicit inherently does not contain direct information about users whether they liked or disliked an item but engineers can infer statements from the dataset. In addition, they have to handle the weight of positive and negative class examples. Positive refers to the likes and negative refers to the dislikes. As explicit feedback is sparse, in implicit feedback, there are way more samples in the negative class rather than the positive. Hence, it is important to use some sort of sampling technique to reduce the undesired impacts of negative samples on the performance of the model.

A. Negative Sampling

When using implicit feedback, different techniques exist in negative sampling. There are essentially two main categories which are Heuristic Negative Sampling (HNS) and Model-based Negative Sampling algorithms. Under HNS, Random Negative Sampling (RNS) and Popularity-biased Negative Sampling (PNS) [14] take place. RNS is simple and easy to implement, as it picks samples randomly and does not consider any weight information. Unlike RNS, PNS weights the samples depending on their popularity and this increases the amount of information on negative samples [15]. There is a method called Weighted Regularized Matrix Factorization (WRMF) [16] which weights and regularizes the matrix factorization process. It uses a coefficient as a regularization term. To sum up the negative samples' impact, different techniques are used to reduce the impact of unobserved or negative samples in the training period.

B. SVD++ and timeSVD++

An improvement of SVD, which is called SVD++ is a method to include the effect of implicit information. In the name, the '++'

part means incorporating implicit feedback [17]. The paper presents SVD++ in detail with corresponding formulas. This method may work well on a combination of explicit and inferred implicit datasets. Another version of SVD is called timeSVD++, where the bias and factor terms are a function of time, i.e. they change over time. This SVD version allows using time information in the recommender. The context information in a recommender may be critical since users' behaviours change over time, location or even their mood.

VII. Loss Types and Learning to Rank

Different objective functions may be used in RS domain and there are different loss types with respect to their properties. Some of the loss types can be listed below.

- Least Squares is the loss type where we treat the problem as regression. When the errors are assumed to be normally distributed, least squares can be derived as the Maximum Likelihood Estimate (MLE).
- Binary Cross-Entropy is a special class of cross-entropy, where the target class of the prediction is 1 or 0 [18] and is taken place in binary classification problems. As Logistic Regression, a sigmoid function is used. When the predictions are assumed to be a Bernoulli distribution, binary cross-entropy can be derived as the negative log-likelihood [19].
- Multi-Label Cross-entropy is another type of cross-entropy which is used in multi-label classification problems. The predictions are assumed to be a multi-nomial distribution. This type is generally implemented for language models.

A. Learning to Rank

L2R or LtR is a method to rank or sort the candidate items in RS according to relevance to users, i.e. selecting top-n items based on their scores from predictions. Apart from RS, it is implemented in search engines since they aim to find the most relevant results depending on the searched keyword. Also, they are generally utilized when the dataset is implicit. Three different kinds can be listed below.

- Point-wise considers the individual items when ranking.
- Pair-wise takes pairs of items instead of individuals to better fit in user preferences. An example method of this type is Bayesian Personalized Ranking (BPR) [20].
- List-wise considers lists of items and it specifically focuses on improving some metrics such as Mean Average Precision (MAP), and Mean Reciprocal Rank (MRR) [21]. Specific use cases are exploration and exploitation methods from reinforcement learning [22].

VIII. Evaluation Metrics

Different evaluation metrics should be used to better assess the performance of the recommender.

Accuracy is the fraction of correct recommendations out of the total possible recommendations [23]. It is one of the most widely used metrics in classification problems including RS domain. The formula of accuracy in RS can be written as Equation 3 [24].

$$accuracy = \frac{number\ of\ successful\ recommendations}{number\ of\ recommendations} \tag{3}$$

Mean Absolute Error (MAE), measures the average absolute deviation between each predicted rating and each user's real values. In Equation 4, u is a user and i is an item while p denotes the ground truth values and P represents the estimated values.

$$MAE = \frac{\sum_{u,i} |p(u,i) - P(u,i)|}{N} \tag{4}$$

Accuracy and MAE's relationship can be shown as in Equation 5.

$$MAE = 1 - accuracy \tag{5}$$

There are other classification metrics which are used in the evaluation of a recommender such as *precision*, *recall* and *F1-score* which is the harmonic mean of precision and recall. Receiver Operating Characteristics (ROC) is another metric which is successful when performing comprehensive assessments. It plots true positive against false positive rates. There are also some metrics such as Mean Average Precision for top-k (MAP@K) and

Mean Average Recall for top-k (MAR@K) where top-k items are selected from the item space based on their prediction scores.

Long Tail Plot is a graph which shows the popular and non-popular items distribution. It places the volume of the popular items in the first part of the graph and the second part gives information about the non-popular items where the curve looks like a tail. Figure 3 illustrates an example which was plotted in the practical session of the course. This plot may be interesting because developers and relevant stakeholders can focus on the non-popular part and increasing the number of interactions between users and non-popular items may allow better user satisfaction thanks to new tastes.

A special metric in RS is *coverage* which measures the fraction of items in the search space where the system is able to provide recommendations. It is about recommending the items which were used in the training.

Personalization, is one of the important aspects of a recommender since we want it to recommend personalized items. On the other hand, *novelty* describes how popular the recommended items are, which is the opposite of personalization.

Another metric is *serendipity*, which tells how the recommendations surprise users in a good way. If a model recommends out-of-preference items to users and if the users like them, this means the model performs well when considering serendipity.

Persistence means recommending the same item to a user more than one time, which is in fact natural requirement in some domains such as e-commerce. Because companies would want their customers to purchase their products as many as they like.

IX. Advanced Topics in RS

Some additional topics in RS which are important to discuss will be covered in this section.

A. Cold Start - Details

The definition of cold start problem and its occurrence were mentioned before in Section II. Apart from being a new user in the system, a user who changes their preferences a lot in a short amount of time may confuse the recommender and cause cold start problem. For the item side, since it takes time for non-popular items to be interacted with users, these non-popular items may arouse cold start as well.

The nature of users in an RS environment shows that their preferences change over time and other conditions. Also, when there are millions of items, there will always be some items which face the least interaction rate. After mentioning the additional reasons for cold start problem, all these reasons clarify that cold start problem is difficult to be completely solved, i.e it may always persist up to a level. The cold start problem is a natural problem of recommenders.

To overcome this, different ways exist. For a new user in the system, the recommender may directly recommend the popular items or it may use the contextual information of new users as they would provide it already. For new items, instead of CF model, a Content-based model would perform well using the characteristics of items without their interaction requirement.

B. Incorporate User Preferences

Incorporation of user preferences refers to the use of information about the preferences, interests, and behaviours of individual users to personalize the recommendations made by the system. To better explain the concept, two feature types should be discussed: objective and subjective features. Objective features of items are their special attributes which describe them or give information in detail. However, subjective feature means the features that the user specifies in a system. For instance, for a song streaming website, the properties of a song such as the singer's name, release date, and genre are the objective features. On the other side, a listener's preferences, filters or settings are subjective features because the listener provides these to have a personalized experience. The incorporation of these two types of features when making recommendations may result in better user satisfaction. In the study of [25], scientific paper recommendation is the domain of the work. The authors' proposed a method which incorporates objective and subjective features. The method is entirely unsupervised and can be applied to any collection of publications.

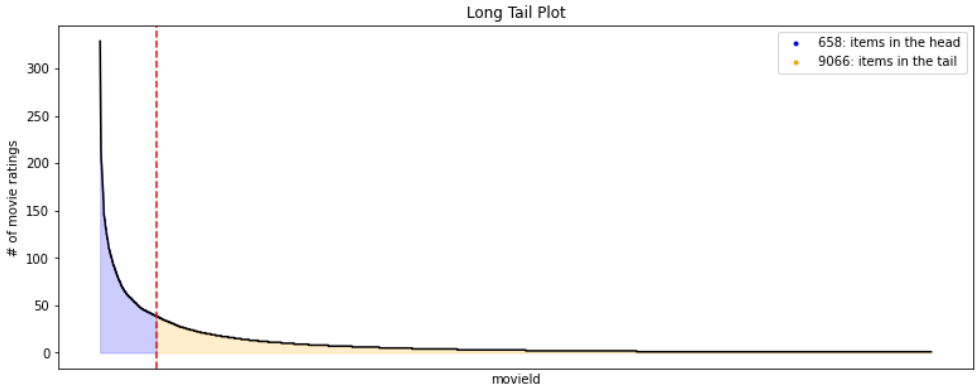

Fig. 3. Long Tail Plot

X. Context-Aware RS

So far, different methods to build a recommender were taken place in the study. A type of RS which is driven by contextual information of users is Context-aware RS. As it was mentioned in Section VI, the context information adds different information to the system and it may improve the performance. Users's location, device, operating system, date and time are examples of context. Context has two different kinds: representational and interactional. Representational context information is observable and does not change over time while interactional information's scope changes over time as it interacts.

There are three ways how to add context information in the recommendation process which are listed below.

1) Pre-filtering refers to adding the context information before the training of the algorithm and using them as direct filters on the data.
2) Post-filtering includes the context information after the training and making recommendations. This can be thought of as directly filtering the recommended items.
3) Modeling is the way where context information is added to the training process and used right in the modeling step.

A diagram of these three ways is shown in Figure 4. The study [26] presents Context-aware RS and the authors inspire developers to have an idea of combining these three techniques in a way to see if the performance gets better.

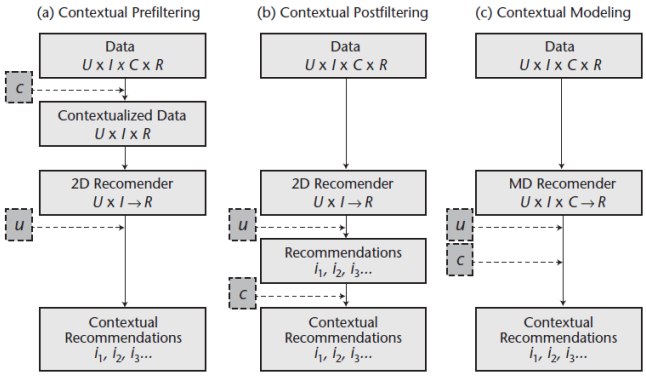

Fig. 4. Ways to Incorporate Context [26]

XI. Block-Chain-Based RS

Block-chain technology is deep and deserves its own study to be discussed. In this section, its advantages and disadvantages in RS domain will take place. Block-chain-based RS is a type of RS that uses a block-chain to store and process data about users, items, and interactions between them [27]. Block-chain technology has key characteristics, such as decentralisation, persistency, anonymity and auditability [28].

A general flowchart of block-chain-based RS can be viewed in Figure 5. There have been proposed methods based on the general workflow, for instance, [29] introduces a block-chain-based RS with a decentralized ledger which includes block-chain and Peer-to-Peer (P2P) network. The two main elements of the flowchart are block-chain block and the recommender. The company/RS owner gathers and stores other useful data for the

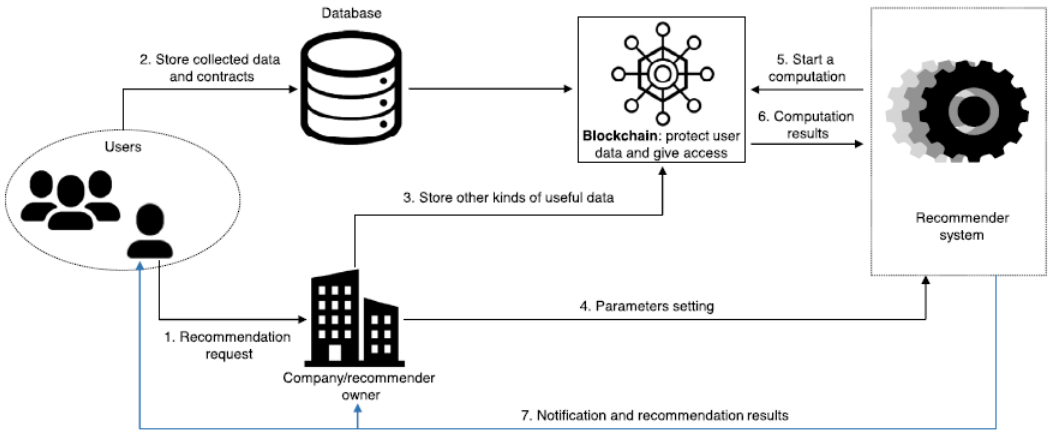

Fig. 5. General Flowchart of Block-Chain-Based RS [27]

RS in the blockchain network. Then, some parameter settings are done followed by a computation step for recommendations. In the end, notifications are pushed to users for recommendations.

Some advantages of this type of RS can be listed below.

- Datasets for RS may contain sensitive information and block-chain provides high security.
- Risk reduction techniques have been used in previous studies but were not completely successful. Hence, block-chain may be a suitable option.
- Block-chain in RS may preserve security and privacy which are two significant keys and were not completely considered and handled by the previous methods.
- This technology allows for a decentralized and transparent system, where data is stored and processed by multiple parties rather than being controlled by a single entity.
- It can help to increase trust and accountability in RS.

Disadvantages can be discovered below with respect to their key aspects. [27]

- Complexity: Block-chain can be complex and require a steep learning curve, i.e. longer and more complex training process may be required.
- Performance: It may not perform as well as traditional centralized systems in terms of speed and scalability.
- Data privacy: While it can provide a high level of security, it may not be suitable for sensitive personal or financial information.
- Regulatory issues: Since it is new, there might be regulatory issues or uncertainty about its use in certain industries or countries.
- Limited adoption: It may be difficult to find developers with expertise in block-chain programming.

XII. Conclusion

This research provided an overview of the different types of RS, including content-based, collaborative, and hybrid systems, as well as the challenges and limitations that these systems face. Additionally, advanced topics such as context awareness and blockchain technology in RS with their potential impact on the field were taken place. As the amount of available data continues to grow and the demand for personalized recommendations increases, it is likely that the field of RS will continue to evolve and improve. However, it is important to consider that there are also limitations and challenges that need to be addressed, such as data privacy, scalability and real-time processing issues.

I think some potential research topics can be building a recommender using Deep Learning (DL), e.g Neural Collaborative Filtering [30] or capturing data not only the classical databases but also from the Internet of Things (IoT) devices such as smartphones and self-driving cars. In my

opinion, a research question could be if there is an efficient way to build a recommender using Transformers [31], which achieved remarkable success in many Artificial Intelligence (AI) fields, such as Natural Language Processing (NLP), Computer Vision, and audio processing. The study pointed out that the combination of more than one technique as hybrid versions of the techniques outperforms the single approaches due to the utilization of each technique's strengths. Further studies will be conducted about the new and non-experimented topics or methods in RS field beyond the fundamentals by highly considering the hybridization strategy.

REFERENCES

[1] Ricci, Francesco, Lior Rokach, and Bracha Shapira. "Recommender systems: introduction and challenges." Recommender systems handbook. Springer, Boston, MA, 2015. 1-34.

[2] Mandal, Supriyo, and Abyayananda Maiti. "Explicit feedbacks meet with implicit feedbacks: a combined approach for recommendation system." International Conference on complex networks and their applications. Springer, Cham, 2018.

[3] Aggarwal, C. C. "Recommender Systems: The Textbook". Springer 2016. ISBN 978-3-319-29657-9

[4] Linden, Greg, Brent Smith, and Jeremy York. "Amazon. com recommendations: Item-to-item collaborative filtering." IEEE Internet computing 7.1 (2003): 76-80.

[5] Ekstrand, Michael D., et al. "Rethinking the recommender research ecosystem: reproducibility, openness, and lenskit." Proceedings of the fifth ACM conference on Recommender systems. 2011.

[6] Ricci, F., Rokach, L., Shapira, B., Kantor, P.B., Recommender Systems Handbook (2011).

[7] Susan (2020) Dining in the US today: Eating out(side), Medium. Medium. Available at: https://medium.com/@pigletslee/dining-in-the-us-today-eating-out-side-11f89bbe4841 (Accessed: January 15, 2023).

[8] Lops, Pasquale, et al. "Trends in content-based recommendation." User Modeling and User-Adapted Interaction 29.2 (2019): 239-249.

[9] Wikipedia contributors. (2020, March 7). Knowledge-based recommender system. Wikipedia. https://en.wikipedia.org/wiki/Knowledge-based_recommender_system

[10] Sridevi, M., and R. Rajeswara Rao. "Decors: A simple and efficient demographic collaborative recommender system for movie recommendation." Adv. Comput. Sci. Technol 10.7 (2017): 1969-1979.

[11] Cunningham, Padraig, and Sarah Jane Delany. "K-nearest neighbour classifiers-a tutorial." ACM Computing Surveys (CSUR) 54.6 (2021): 1-25.

[12] Fan, Wenqi, et al. "Deep social collaborative filtering." Proceedings of the 13th ACM Conference on Recommender Systems. 2019.

[13] Rendle, S., Freudenthaler, C., Gantner, Z. and Schmidt-Thieme, L., 2009, June. BPR: Bayesian personalized ranking from implicit feedback. In Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence (pp. 452-461). AUAI Press

[14] Chen, Ting, et al. "On sampling strategies for neural network-based collaborative filtering." Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 2017.

[15] Jiang, J. (2021) Overview negative sampling on recommendation systems, Medium. MLearning.ai. Available at: https://medium.com/mlearning-ai/overview-negative-sampling-on-recommendation-systems-230a051c6cd7

[16] Gu, Quanquan, Jie Zhou, and Chris Ding. "Collaborative filtering: Weighted nonnegative matrix factorization incorporating user and item graphs." Proceedings of the 2010 SIAM international conference on data mining. Society for Industrial and Applied Mathematics, 2010.

[17] Koren, Yehuda. "Factorization meets the neighborhood: a multifaceted collaborative filtering model." Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining. 2008.

[18] Ruby, Usha, and Vamsidhar Yendapalli. "Binary cross entropy with deep learning technique for image classification." Int. J. Adv. Trends Comput. Sci. Eng 9.10 (2020).

[19] Mehta, Pankaj, et al. "A high-bias, low-variance introduction to machine learning for physicists." Physics reports 810 (2019): 1-124.

[20] Rendle, Steffen, et al. "BPR: Bayesian personalized ranking from implicit feedback." arXiv preprint arXiv:1205.2618 (2012).

[21] Shi, Yue, et al. "Climf: learning to maximize reciprocal rank with collaborative less-is-more filtering." Proceedings of the sixth ACM conference on Recommender systems. 2012.

[22] Karatzoglou, Alexandros, Linas Baltrunas, and Yue Shi. "Learning to rank for recommender systems." Proceedings of the 7th ACM Conference on Recommender Systems. 2013.

[23] Isinkaye, Folasade Olubusola, Yetunde O. Folajimi, and Bolande Adefowoke Ojokoh. "Recommendation systems: Principles, methods and evaluation." Egyptian informatics journal 16.3 (2015): 261-273.

[24] Del Olmo, Félix Hernández, and Elena Gaudioso. "Evaluation of recommender systems: A new approach." Expert Systems with Applications 35.3 (2008): 790-804.

[25] Igbe, Tobore, and Bolanle Ojokoh. "Incorporating user's preferences into scholarly publications recommendation." Intelligent Information Management 8.02 (2016): 27.

[26] Adomavicius, Gediminas, and Alexander Tuzhilin. "Context-aware recommender systems." Recommender systems handbook. Springer, Boston, MA, 2011. 217-253.

[27] Himeur, Yassine, et al. "Blockchain-based recommender systems: Applications, challenges and future opportunities." Computer Science Review 43 (2022): 100439.

[28] Zheng, Zibin, et al. "Blockchain challenges and opportunities: A survey." International journal of web and grid services 14.4 (2018): 352-375.

[29] Lisi, Andrea, et al. "A smart contract based recommender system." International Conference on the Economics of Grids, Clouds, Systems, and Services. Springer, Cham, 2019.

[30] He, Xiangnan, et al. "Neural collaborative filtering." Proceedings of the 26th international conference on world wide web. 2017.

[31] Lin, Tianyang, et al. "A survey of transformers." AI Open (2022).