December 12th, 2021

Personal Project : Evolution of Collaborative Filtering Algorithms with Implicit Feedback For Large Scale Recommended Systems

DATA 598

ROHIT LOKWANI

1. Introduction

A recommender system, or a recommendation system, is a subclass of information filtering systems that seeks to predict the "rating" or "preference" a user would give to an item [9]. Recommendation systems work in two broad categories: Collaborative filtering and Content-based filtering. The former depends on clusters of user data and the latter on user-specific engagement. We focus on the latter which has a primary aim to analyze user data to provide personalized recommendations [10]. In our essay, we'll focus on work done in this domain by companies like Spotify, Netflix, Google, amongst others to learn how these applications work on a large scale.

Starting with Spotify as an example. Traditionally, Spotify has relied primarily on collaborative filtering approaches for their recommendations. This works well for them, as it involves determining user preference from historical behavioral data patterns. An example of this is if two users listen to the same sets of songs or artists, their tastes are likely to align. We talk more about this in the next section. Christopher Johnson, former Director of Data Science at Spotify, worked on the launch of Discover Weekly. According to Johnson, a Content-Based strategy relies on analyzing factors and demographics that are directly associated with the user or product, such as the age, sex, and demographic of the user or a song genre or period, such as music in the 70's or 80's [6]. Recommendation systems that are based on Collaborative Filtering take consumer behavior data and utilize it to predict future behavior [6]. This consumer behavior leaves a trail of data, generated through implicit and explicit feedback. Spotify relied primarily on implicit feedback to train its algorithm. Examples of user data based on implicit feedback can be playing a song on repeat or skipping it entirely after the first 10 seconds. User data is also gleaned from explicit feedback such as the heart button on Discover Weekly or songs that were liked that automatically save in the library and the "Liked from Radio" playlist. An example of the myriad other ways in which collaborative filtering and recommendation algorithms work in different approaches is evident in the diagram below (see Figure 1). Spotify uses a combination of 4 approaches – Attribute-based, CF (item by item), CF (user similarity) and Model-based.

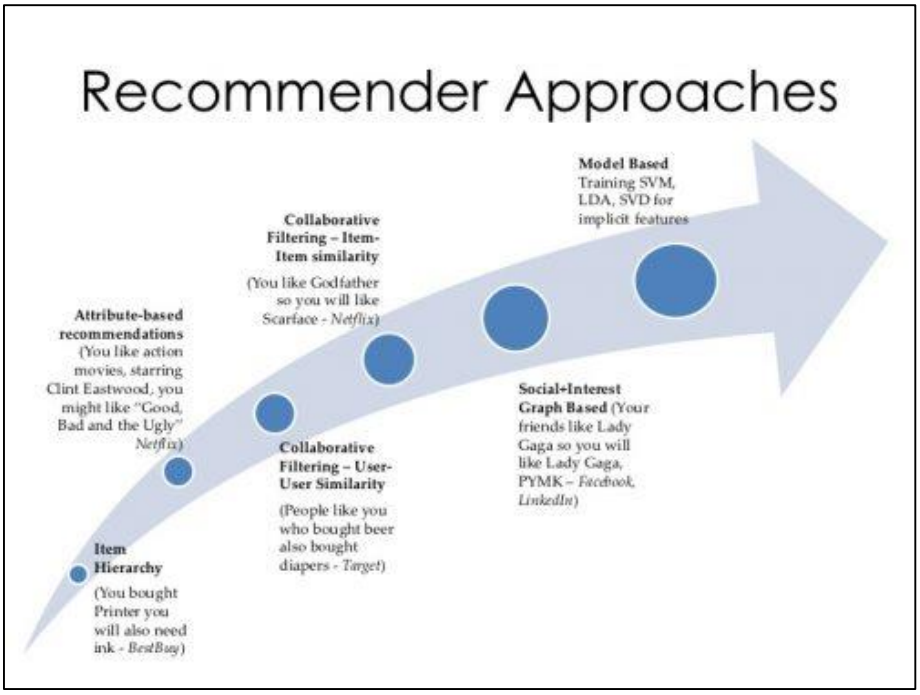

Figure 1: Different Recommender Approaches [11]

In further sections, we focus on the previous work done in the field of Collaborative filtering using implicit feedback, followed by learnings from state-of-the-art papers, challenges with algorithms and how other algorithms help overcome them, and some new suggestions that can further improve them.

2. Background Summary / Previous Research:

Here, we summarize three different types of algorithms as a background summary of work done in this domain. Clustering is used at Spotify to develop a user taste profile, statistical methods like correlation to cluster users together, and the third one using deep neural networks at YouTube to rank recommendations according to the user. Since neural networks are our prime focus, we focus on them more in the latter half of this section.

2.1 Clustering at Spotify [5]

Spotify uses ML in three different ways: Collaborative filtering, NLP, and Audio modeling. Collaborative filtering used for the User-specific Discover Weekly playlist in 2015 led to a boost in Spotify's revenues [5].

In terms of Spotify, Discover Weekly and other playlists are created using collaborative filtering, based on the user's listening history, in tandem with songs enjoyed by users who seem to have a similar history. Additionally, Spotify uses "Taste Analysis Data" to establish a Taste Profile. This technology, groups the music users frequently listen to into clusters and not genres, as the human categorization of music is largely subjective. Examples of this are evident in Spotify's Discover Weekly and Daily Mix playlists suggestions. Clustering algorithms like Spotify's group data are based on their similarities. This clustering is described as *an* "exploratory data analysis technique where we identify groups naturally occurring in the data" [5]. The figure below describes the process.

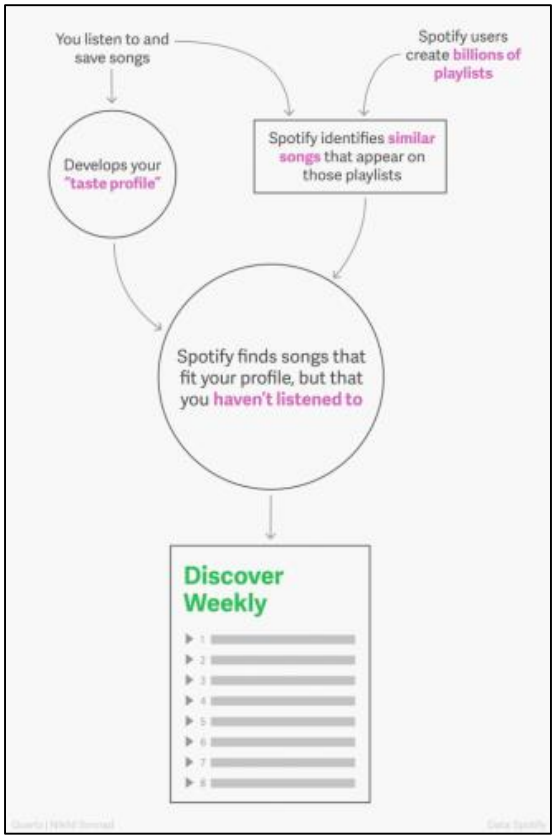

Figure 2: **Taste Profile Selection Process** [12]

2.2 Memory-based Collaborative Filtering [4]

In this paper[4] the authors followed a memory-based approach to recommend songs for the user. They used Pearson's correlation coefficient between users to calculate their similarity based on the historical songs listened to.

Once the subset of most similar users to the active user is obtained, the last step is to predict what items the active user might be interested in using the weighted average of all the ratings, defined in the following equation:

$$P_{a,i} = \overline{r_a} + \frac{\sum_{u \in K} (r_{u,i} - \overline{r_u}) \times w_{a,u}}{\sum_{u \in K} w_{a,u}}.$$

Where:

P (a,i) is the prediction for the active user for the item,
W(a,u) is the similarity between users a and u,
K is the subset with the most similar users,
r(u,i) is the user's rating u for the item i,
r(u) is the average rating of the user u.

They also experiment with Phi coefficient to understand the difference in performance. Although Pearson's correlation performs better, Phi's coefficient helps in reducing the latency and is hence preferred by the authors.

2.3 Deep Neural Networks for YouTube Dashboard Recommendations [2]

In this paper, the model is divided into two modules:

1. Candidate Generation Network, which takes users' historical data and then suggests videos from a large corpus using collaborative filtering. The similarity between users is expressed in terms of coarse features such as IDs of video watches, search query tokens, and demographics. The candidate generation model is an application of extreme multiclass problems, wherein they take the user's context as input features and classify each video as relevant or not. This method is based on explicit feedback. They speak about the correct set of activations used to reduce latency.

2. The ranking network assigns a score to each video according to the desired objective function using a rich set of features describing the video and user. The highest scoring videos are presented to the user, ranked by their score. The primary role of ranking is to use impression data to specialize and calibrate candidate predictions for the particular user interface. The ranking is done using logistic regression.

Their original matrix factorization method just had shallow networks which used users' previous watches. Now, it's more of a non-linear generalization of factorization techniques. Deep neural networks require special representations of categorical and continuous features which they transform with embeddings and quantile normalization, respectively. Layers of depth were shown to effectively model non-linear interactions between hundreds of features. Logistic regression was modified by weighting training examples with watch time for positive examples and unity for negative examples, allowing them to learn odds that closely model expected watch time. This approach performed much better on watch-time weighted ranking evaluation metrics compared to predicting click-through rate directly.

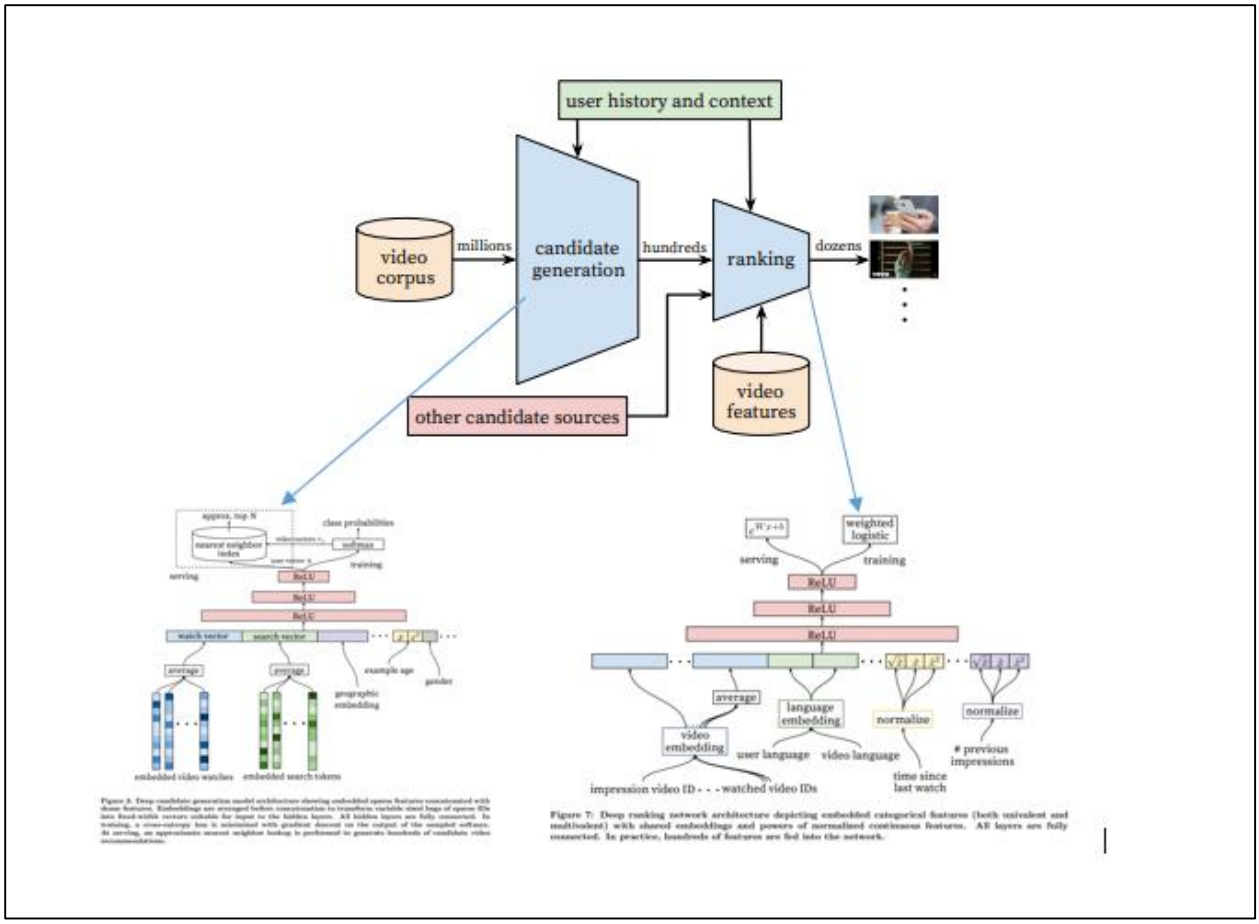

Figure 3: Candidate Generation and Ranking Architecture [2]

3. Summary of State-of-art Algorithms:

In this section, we focus on some of the most widely used state-of-the-art recommender algorithms in recent years.

3.1 Logistic Matrix Factorization [6]

Spotify further analyzes and applies user data by using a matrix decomposition method, which is also known as matrix factorization. The approach of matrix factorization aims to find answers by 'decomposing' (hence the term matrix decomposition) the data into two separate segments[6]. The first segment defines the user in terms of marked factors, each of which is weighted differently. The second segment maps between factors and products, which in the Spotify universe are songs, artists, albums and genres, thus defining a factor in terms of the products offered. Basically, each customer has only heard a small percentage of the songs and the overall songs have only been watched by a small percentage of the customers. Based on these assumptions, the learning algorithm needs to be able to generalize and predict successfully.

Each row of the data matrix X contains songs. Most of this data, however, is missing, as the customer has not yet watched many of the movies, which is where the recommendation system comes in. The matrix then factors this into two splits – F and G – where F is factors and G movies/music. Spotify uses a matrix factorization application called Logistic Matrix Factorization or Logistic MF[6], to generate lists of related artists, for example for Artist 'Radio' playlists, based on binary preference data. This matrix is established by calculating millions of recommendations based on millions of other user behavior and preferences

The challenges with matrix factorization include: First, the number of parameters in any matrix factorization model is huge: it linearly depends on the number of both users and items, which leads to slow model learning and overfitting. Second, to make a prediction for a new user/item based on their ratings, one has to run an optimization procedure in order to find the corresponding user/item embedding. Third, only a small amount of ratings are known for some (often for a majority of) users and items, which could also lead to overfitting. This makes it necessary to heavily regularize matrix factorization models, and standard L1 or L2 regularizers are hard to tune. [3] The Variational Autoencoder for Collaborative Filtering (Mult-VAE) [8] is a subsequent improvement of CDAE that extends it to multinomial distributions in the likelihood, which are more suitable for recommendations.

3.2 Variational Autoencoders For Collaborative Filtering [8]

A variational autoencoder can be defined as being an autoencoder whose training is regularised to avoid overfitting and ensure that the latent space has good properties that enable the generative process.

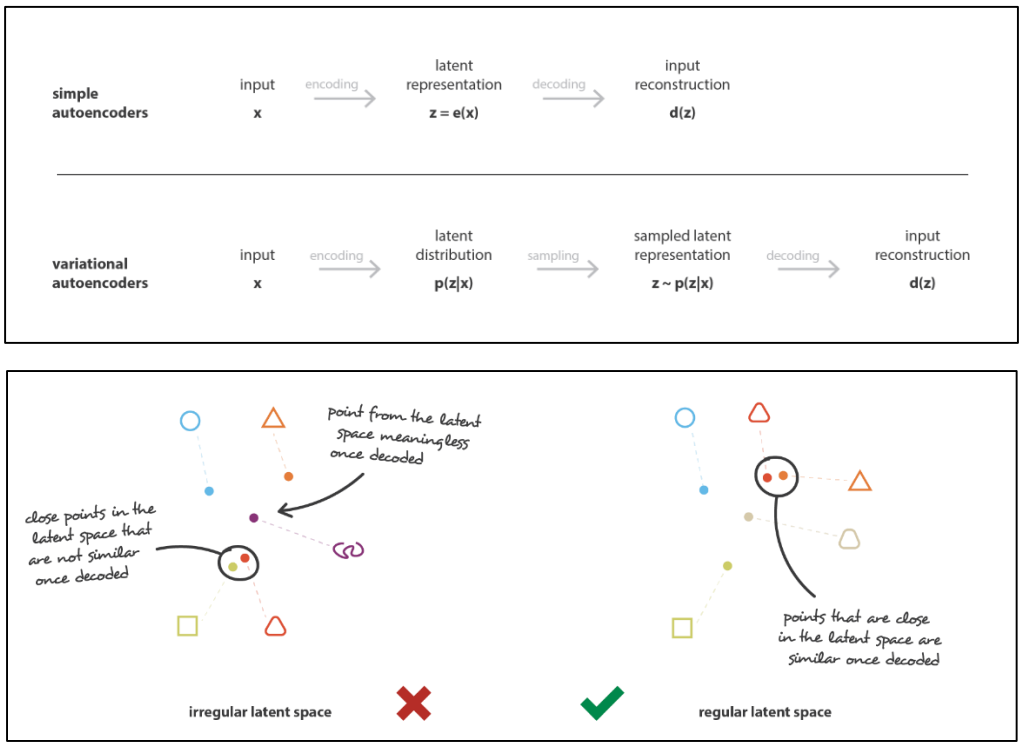

Figure 4: Understanding Variational Autoencoders (VAEs) [13]

In the paper [8], first, they use a multinomial likelihood for the data distribution. They showed that this simple choice realizes models that outperform the more commonly used Gaussian and logistic likelihoods. Second, they reinterpreted and adjusted the standard VAE objective, which they argued was over-regularized. They then draw connections between the learning algorithm resulting from their proposed regularization and the information-bottleneck principle and maximum-entropy discrimination.

Based on an alternative interpretation of the VAE objective, we introduce an additional regularization parameter to partially regularize a VAE (Mult-VAE). They also provide a practical and efficient way to tune the additional parameter introduced using KL annealing. Mult-VAE significantly outperforms the state-of-the-art baselines on several real-world datasets, including two recently proposed neural-network-based approaches. Finally, they identify the pros and cons of both Mult-VAE and denoising autoencoders and show that employing a principled Bayesian approach is more robust.

3.3 RecVAE: A New Variational Autoencoder for Top-N Recommendations with Implicit Feedback [3]

In this work from Samsung Research, they proposed a Recommender VAE (RecVAE) model for collaborative filtering with implicit feedback based on the variational autoencoder (VAE) and specifically on the Mult-VAE approach. RecVAE presents several important novelties that together combine into significantly improved performance. First, they designed a new architecture for the encoder network. Second, they have introduced a novel composite prior distribution for the latent code z in the variational autoencoder. The composite prior is a mixture of a standard Gaussian prior and the latent code distribution with parameters fixed from the previous iteration of the model an idea originating from reinforcement learning where it is used to stabilize training. In the context of recommendations, they have also found that this prior improves training stability and performance. Thirdly, they have developed a new approach to setting the hyperparameter β for the Kullback-Leibler term in the objective function. They have found that β should be user-specific, $\beta = \beta(xu)$, and should depend on the amount of data (implicit feedback) available for a given user. Finally, they introduce a novel approach for training the model. In RecVAE, training is done by alternating updates for the encoder and decoder. This approach has two important advantages. First, it allows performing multiple updates of the encoder (a more complex network) for every update of the decoder (a very simple, single-layer network that contains item embeddings and biases). Second, it allows using of corrupted inputs (following the general idea of denoising autoencoders) only for training the encoder while still training the decoder on clean input data. This is again beneficial for the final model training due to the differing complexities of the encoder and decoder. As a result of the above novelties, their model significantly outperforms all autoencoder-based previous works and shows competitive or better results in comparison with other models across a variety of collaborative filtering datasets, including MovieLens-20M (ML-20M), Netflix Prize Dataset, and Million Songs Dataset (MSD)

3.4 Explore, exploit, and explain: Personalizing Explainable Recommendations with Bandits [1]

This paper uses a reinforcement learning approach where Exploitation recommends content with the highest predicted user engagement and has been the focus of traditional recommender systems. Exploration gathers more information. Plus here, they combine recsplanations (explaining recommendations) in a principled manner. The specific algorithm that Spotify uses is an epsilon-greedy solution for the multi-armed bandit problem. However, in the face of uncertainty from only a small amount of data, the recommender will sometimes suboptimally ignore relevant items. The fact that the recommender itself is active in deciding its training data perpetuates this problem.

In summary, their contributions include: They identify and provide a formalization for the problem of jointly personalizing explainable recommendations. They present Bart, a contextual bandits-based framework that addresses the problem of recommending explanations under uncertainty of user satisfaction. They implement Bart in a scalable production environment presented challenges that they discuss and address. Through offline experiments on randomized historical data and online experiments with users on the homepage of a large-scale music recommender system, they empirically evaluate Bart and find that it significantly outperforms the best static ordering of explanations

4. Other Challenges with Recommender Systems:

A couple of the challenges with these algorithms include– Scale, Freshness, and Noise. Freshness - New users cannot easily be a part of the recommendations. Here is when the Multi-armed Bandit approach is useful and here is when Spotify uses the audio models to understand the audio and recommend according to the genre[1]. Another one is a cold start problem where when you add new items they are not recommended easily, which is dealt with Variational Autoencoder approaches[8]. Another is a Surrogate problem and its solution developed in the Youtube paper[2].

5. Conclusion

After going through all of these papers, some ideas that I can think of is currently Spotify uses recommendations based on audios, it has an inbuilt feature that shows videos of the songs as well. One way could be utilizing algorithms used by YouTube/Netflix to suggest songs that the user has seen videos for and further improve predictions. A Hybrid Variational Autoencoder for Collaborative Filtering can be used for multimodal learning and predictions.

In a nutshell, in tandem with advances in technology and media affordances, future implications of machine learning include more personalized, immersive user experiences with progressively complex features. With recommendation algorithms choosing what content we watch, what we listen to guiding users towards certain choices and might end up ‘ pigeonholing’ users. Hence, a balance of exploration, exploitation, and Multi-modal input is a must. It is important to remember, however, that ultimately these algorithms are trained and designed. Despite the often hyperbolic coverage it receives, the overarching umbrella of AI in and of itself relies heavily on machine learning and ML fairness. However, in today’s world of fluid musical

genres and especially while applying the concepts of pattern recognition, machine learning, and collaborative filtering, most of the user-generated data is still subjective – a microcosm of the larger sociotechnical system we live in [5].

6. References

[1] McInerney, J., Lacker, B., Hansen, S., Higley, K., Bouchard, B., Gruson, A. & Mehrotra, R. (2018). Explore, exploit, and explain: personalizing explainable recommendations with bandits. Proceedings of the 12th ACM Conference on Recommender Systems (RecSys '18). Association for Computing Machinery, New York, NY, USA, 31–39. https://doi.org/10.1145/3240323.3240354

[2] Covington, P., Adams, J. & Sargin, E. (2016). Deep Neural Networks for YouTube Recommendations. Proceedings of the 10th ACM Conference on Recommender Systems (RecSys '16). Association for Computing Machinery, New York, NY, USA, 191–198. https://doi.org/10.1145/2959100.2959190

[3] Shenbin, I., Alekseev, A., Tutubalina, E., Malykh, V. & Nikolenko, S. I. (2020). RecVAE: A New Variational Autoencoder for Top-N Recommendations with Implicit Feedback. Proceedings of the 13th International Conference on Web Search and Data Mining (WSDM '20). Association for Computing Machinery, New York, NY, USA, 528–536. https://doi.org/10.1145/3336191.3371831

[4] Pérez-Marcos J. & López Batista V. (2018) Recommender System Based on Collaborative Filtering for Spotify's Users. Trends in Cyber-Physical Multi-Agent Systems. The PAAMS Collection - 15th International Conference, PAAMS 2017 (PAAMS '17). Advances in Intelligent Systems and Computing, vol 619. Springer, Cham. https://doi.org/10.1007/978-3-319-61578-3_22

[5] Huq, P. (2019, May 6). *Music To My Ears: De-Blackboxing Spotify's Recommendation Engine | CCTP-607: "Big Ideas": AI to the Cloud*. Center for New Designs in Learning and Scholarship. Retrieved December 11, 2021, from https://blogs.commons.georgetown.edu/cctp-607-spring2019/2019/05/06/music-to-my-ears-de-blackboxing-spotifys-recommendation-algorithm/

[6] Christopher J. Logistic Matrix Factorization for Implicit Feedback Data. https://web.stanford.edu/~rezab/nips2014workshop/submits/logmat.pdf

[7] Daniel R. (2020, December 6). *How Spotify recommender system works.* https://www.linkedin.com/pulse/how-spotify-recommender-system-works-daniel-roy-cfa/

[8] Dawen D., Rahul G. K., Matthew D. H., & Tony J. 2018. Variational Autoencoders for Collaborative Filtering. In Proceedings of the 2018 World Wide Web Conference (WWW '18). International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 689–698. DOI:https://doi.org/10.1145/3178876.3186150

[9] Francesco R., Lior R. & Bracha S. (2011) Introduction to Recommender Systems Handbook, Recommender Systems Handbook, Springer, pp. 1-35

[10] Clark B. (2019, November 11). *How Spotify Recommends Your New Favorite Artist.* https://towardsdatascience.com/how-spotify-recommends-your-new-favorite-artist-8c1850512af0

[11] Yong Z. (2015, March 4). *Matrix Factorization in Recommender Systems.* https://www.slideshare.net/irecsys/matrix-factorization-in-recommender-systems

[12] Adam P. (2015, December 15). *The magic that makes Spotify's Discover Weekly playlists so damn good*. https://qz.com/571007/the-magic-that-makes-spotifys-discover-weekly-playlists-so-damn-good/

[13] Joseph R. (2019, December 23). *Understanding Variational Autoencoders (VAEs)* https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73