Recommendation System: RockSmith (Ubisoft)

Application of SVD in Collaborative Filtering

Seyoung Oh
M.S. Data Science: GalvanizeU, University of New Haven
44 Tehama St. San Francisco, CA
armyohse@gmail.com

Abstract – Potent gaming program designs are underpinned by equally sound user individualization and predisposition. Effective applications such as machine learning library have been facilitative in highlighting leads that carry the most potential. In this project, we will be building a recommendation system known as Rocksmith, which is developed on Ubisoft and can be used in music playlists to suggest and navigate from one song to another. The SVD suggestion configuration was tested on a selected Ubisoft data team and the outcomes were encouraging when measured against various classification benchmarks. The most revealing indicators vis-à-vis user-oriented individualization were song start and stop, musical varieties, timelines and user playlists. Relative success was achieved using Tensorflow, which is one of the most impactful GPU-aligned deep learning tools.

I. INTRODUCTION

Needs identification is at the core of the clamor for effective user interfaces and suggestion programs for gaming programming are growing ever more prevalent. Rocksmith is a PC or console-based application that allows users to strum their guitars in gaming mode. As a consequence, it was imperative to concentrate on users' genre preferences. An advanced platform such as Netflix optimizes this technique by embedding film features in their algorithms. However, the applicability of recommendation programs is not limited to films. Success has also been realized in other segments like E-commerce. In spite of this, such areas are not ideal for video games. As a matter of fact, there are numerous web-based gaming databases, but they are not designed to accommodate or provide recommendation services. Because of this, their suggestion and search functions are scant.

Rocksmith is far from being just an average game; in fact, it is also a music player. In this sense, it works by employing gamers' guitars in playing music. In doing this, it guarantees rapid results for both basic and intricate searches by eliminating the time that is often spent ranking songs. One may wonder how this is made possible. Well, the logic lies in the way in which the system's data configurations are dovetailed with over 5,000,000 user playlists. Rocksmith's suggestion algorithms are extremely reliant on data, which explains the decision to prefer Ubisoft despite the availability of document and relational databases. As demonstrated by most data-based successes, the quality of recommendations is directly proportional to the amount of data available. Theoretically, nodes can be classified into two groups – characteristic and game – that both comprise fundamental specifications such as playtime, name and distinct ID. As the name suggests, a characteristic node is descriptive in nature, examples include song attributes such as theme, year of release, artist and genre, among others.

Characteristic nodes are interconnected with all game nodes and in some instances they are linked to other characteristic nodes. Examples include a character that is biased towards a particular genre, a location that features a sub-location and a developer working for a company.

The functionality of recommendations is driven by relationships; they are all enjoined by one relevance property which shows that some relationships are of higher priority compared to others. For instance, two songs that belong to the same genre or are composed by the same artist are more closely related compared to two songs with a similar theme. Using Scikit-Learn Machine Learning to perform complex searches is feasible, but when it comes to SVD algorithms more comprehensiveness is provided by libraries such as Pytorch and Tensorflow.

The key alteration in the model that is being analyzed in the relational data format is that connections between two nodes are collated and their importance is decoded and established, such that the greater the significance the closer it will be to the higher number. The rationale behind this is to enable the system to process shorted path algorithms, whereby the shortest path becomes the most relevant between both nodes. Rocksmith has expanded exponentially in recent times, recording considerable growth in sales and annual revenue since its commercial debut. The objective of Ubisoft is to leverage the conversion of opportunities into higher turnover. Data collection demands extensive preparation and it is vital to focus on characterizations with the most aesthetic appeal.

II. OVERVIEW

A. Problem Statement

At the moment, Rocksmith lacks a user-centric recommender feature. In some cases, users are oblivious of the song they would like to play next. This is where the Ubisoft team, with over 4,700 subscribers and over 11 million songs, comes in; these attributes provide suitable conditions for constructing a recommender system. Most games, especially a majority of web-based gaming platforms, operate customized or recommender

features for each subscriber. Since Rocksmith is more of a guitar-oriented music service than an ordinary game, we lean towards mainstream music streaming platforms like YouTube. Buying new songs is primarily a financial decision that is left to gamers. Currently, major recommendation players such as Netflix and YouTube provide extremely effective customization. Other elements that determine whether or not users buy new songs include preferences pertaining to themes, artists, genre and favorite songs. Taken together, these factors result in a complex decision matrix that is difficult to model.

Ubisoft gathers a variety of data related to potential subscribers, including genre, complexity, artists, length of playtime and song names. Data browsing pointers like song start and stop are monitored and can be archived in databases. Finally, the Rocksmith team surveys user conduct and results. Tensorflow, a deep learning library that is popular on Google, can assimilate the different data in order to appraise the recommendation process. For instance, it can detect that a user's favorite genre in a specific playlist might be related to a different user with an identical musical orientation.

Highly customized playlists will be built into the system to cater to users, a move that could be central to profitability. By classifying playlists according to user playlist habits, Ubisoft is capable of focusing on users who require assistance with songs that are related to their preferences. The efficiency of the Rocksmith team spurs users to play more songs and is central to Ubisoft's long-term success.

B. Theory of the solution.

Matrix factorization model is most appropriate for song recommendation system. It is matrix task of inferring a function from time-invariant metric functions. The function is inferred by user data and their play lists. The Singular-value decomposition is a factorization of a real or complex matrix. Since the outcome of prior user data is known and differentiated into different outcomes, SVD is the preferred model.

Suppose that there are N users, M song lists, and the number of playtimes are integers from 1 to K (K-star scale). Overfitting is a common problem in machine learning. With increasingly complex models, there is a risk that the model learns from the noise of a specific training set and fails to generalize to other data sets. Best practices include separating data into train-test sets, cross-validating appropriately, and using model less prone to overfitting.

Since user rating was not include in the dataset, using play-times by users could be replace as user rating. Also, normalization was needed since every song has different released date. Afterward, the algorithms could possibly gain user rating by number of play-times.

The essential approach to appraisal revolves around the train-test process that is ubiquitous in machine learning. The first step consists of a data set usually composed of an amalgamation of histories and user reviews and perhaps comprising bonus information related to the user and aspects. The second step involves dividing the users within the data set into two categories: the test group and training group. The next challenge is building the recommender model in reference to the data set. This is followed by sequential consideration of the users falling under the test group and the splitting of their acquisitions or reviews into two classes: the target group and the query group. The recommender is assigned the query group designated as a user background and commanded to propose items or forecast the review for the elements found in the target group. Finally, it is assessed based on how much its suggestions or recommendations correspond to those saved in the query group.

C. Prior Literature

Factorization model are very popular in recommendation systems because they can be used to discover latent features underlying the interactions between two different kinds of entitles. Singular-value decomposition is one of the most powerful factorization algorithms. Deep learning is the most powerful techniques to implement large data sets. Tensorflow is a general computation framework using data flow graphs although deep learning is the most important application of it. With Tensorflow, derivative calculation can be done by auto differentiation, which means that you only need to write the inference part. Since Tensorflow has embedding module for application, it is supposed to be a great platform for factorization models as well, even in production.

The singular value decomposition (SVD) is then presented along with some related comments on the numerical determination of rank. A variety of applications of the SVD in linear algebra and linear systems is then outlined. SVD was developed by Erhard Schmidt in 1907. Many of business based on recommendation system use SVD algorithms.

The idea of reducing the dimensionality of a dataset is not limited to the singular value decomposition. While SVDs provide one of the most theoretically grounded techniques for finding features, there are several approximation algorithms that can be used on datasets.

Gene Golub and William Kahan (1965) were among the first to take a business perspective towards SVD algorithms. In 1970, Golub and Christian Reinsch published a variant of the Golub/Kahan algorithms that is still the one most-used today.

III. IMPLEMENTATION

A. Choice of tools

The Python have a correspondence to many common mathematical ideas. Additionally, python also has tools that are extremely helpful in working with machine learning and mathematics problem. Deep learning is leading in machine learning industry last few years already. Furthermore, it is most currently popular paradigm among with data scientist. Particularly, we will introduce Tensorflow which is most popular deep learning library. For implementing it requires GPU based computer system, so we used Google cloud system with NVidia Tesla P40.

B. Choice of data

Ubisoft Rocksmith team acquire many different forms of user's data in form. First, they have more than 11 million play lists from 40 thousand users. This was very exciting and motivate opportunities as a data scientist project because usually more data can lead to better quality of product. This data contains Unique User ID, Song Name, and number of play each song.

Song stop was a second data set. This data contains of when user stop playing the game, the mastery level of each song, and many other information that makes able to observe user's behavior. This was helpful to determine features to build better recommendation systems. Since there was enough playlist from users it was totally possible that we assume number of play time can be replace as user rating. Also, normalization was necessary since each song has different released data so we could add weight this matter to each play list.

Lastly, there was a data for each song. There is about 1000 songs released today and each song has specific genre for its own. Since Rocksmith is not just typical game it allows to users connect their own guitars to the game, this data set was critically important. Based on datasets and characteristic of the game, it would be appropriate using entertainment recommendation system such as Netflix or Pandora.

After locating the data, we proceeded to pre-analyze it to suit our platform. Owing to the fact that we opted to use attributes such as playtime, name of song and user ID, we were constrained in the sense that we could only derive indicators that we wanted to use. To perform this process we used Numpy, a python dataframe component, and Pandas.

In addition to the initial three data sets, other data sets were explored. An initial model was built using the three initial data sets. Based on feature impotencies, User ID, song name, number of playlist were discovered as highly matters. The total of data was about 20GB. Due to determine features were most important process in this project, we have decided to use SVD model for the recommender system.

After initiated sample model, it was obvious that each user has their own preference in their playlists even they do not have many play history. Also, we could generate most popular song that possibly use it for new users. This method is very popular for new users we assume that they preferred to see most popular song in the game.

	Item 1	Item 2	Item 3	Item 4	Item 5
User 1	0	3	0	3	0
User 2	4	0	0	2	0
User 3	0	0	3	0	0
User 4	3	0	4	0	3
User 5	4	3	0	4	0

A matrix of user/item ratings

Fig. 1. A matrix of user/item rating. We will use number of playtimes for rating after normalization which gives a weight on every list.

C. Choice of model

Matrix Factorization is a popular recommendation model developed by Erhard Schmidt in 1907. The coefficient indicates the relationship between the user ID and song name. To do this, we must define a user-song matrix first. The only condition is that they must be time-invariant metric functions. This approach models both users and songs by giving them coordinates in a low dimensional feature space each user and each song has a feature vectors. And each rating (we will use number of playtimes) is modeled as the inner product of the corresponding user and song feature vectors. In other words, we assume there exist a small number of (unknown) factors that determine ratings, and try to determine the values of these factors based on training data. Theoretically, based on the training data, we try to find a low-rank approximation of the user-song matrix.

$$M_{UI} = \begin{pmatrix} r_{00} & \vdots & r_{0m} \\ \cdots & \ddots & \cdots \\ r_{n0} & \cdots & r_{nm} \end{pmatrix}$$

Fig. 2. Matrix Factorization. We assume N=Users, M=Song

$$M_{UI} = U\Sigma V^T \text{ where } U \in \mathbb{R}^{n\times p}, \Sigma \in \mathbb{R}^{p\times p}, V \in \mathbb{R}^{p\times m}$$

*Fig. 3. The SVD(Limiting the computation to the p greatest singular values) allows to define the matrix. The columns of **U** contain the left singular vector, while the rows of transposed **V** contain the right singular vectors. Sigma is a diagonal matrix containing the singular values.*

To build a better prediction model, we should consider determine Root-mean-square deviation (RMSE). It is a frequently used measure of the differences between values (sample and population values) predicted by a model or an estimator and the values observed. The RMSD represents the sample standard deviation of the differences between predicted values and observed values. These individual differences are called residuals when the calculations are performed over the data sample that was used for estimation, and are called prediction errors when computed out-of-sample.

Typically, in machine learning theory, determine the RMSE and adjust parameters to achieve the optimal performance. We will adjust all the parameters through K-fold cross-validation. Since, our algorithms are mixed we need to adjust the dimension of the feature vectors and the regularization weight.

For our implementation, predictions as outcome of collaborative filtering algorithms, we further enhance the performance by first applying the

In this model, we assume that if a rating is determined by a set k user latent factors and k song ones. A latent factor is a variable that we cannot directly observe however, it is possible to estimate. This concept is very important because the theoretical foundation of the whole model is built upon it. Just to remind, we assume that a user rates a generic song according to the gender, age range, education, interests and so forth. At the same time, a song can be virtually split into different components that contribute the overall rating (listen count): brand, features, genre, etc. Even if we expect those latent factors, we

cannot easily determine them, therefore we use a factorization method. The original user-song matrix will be split into two factors: a user-factor matrix and a factor-product matrix. The multiplication removes the latent factors, rebuilding the original user-song matrix.

$$M_{UI_k} = U_k \Sigma_k V_k^T$$

Fig. 4. we assume to have k factors, we can truncate the SVD

$$\begin{cases} S_U = U_k \cdot \sqrt{\Sigma_k} \\ S_I = \sqrt{\Sigma_k} \cdot V_k^T \end{cases}$$

Fig.5. Selecting the top k singular values and the corresponding singular vectors. Predictions can be obtained after defining two auxiliary matrices.

$$\tilde{r}_{ij} = E[r_i] + S_U(i) \cdot S_I(j)$$

Fig.6. The predicted rating for user i and song j. Where the first term is the average rating per user. However, in this example we are interested in finding the top k suggestions for all users, therefore we are not going to consider it as it is a constant after defining the user.

With define vectors for both users and songs, we can estimate the degree to which a user would prefer a song by calculating the cosine distance between the user's and song's vectors.

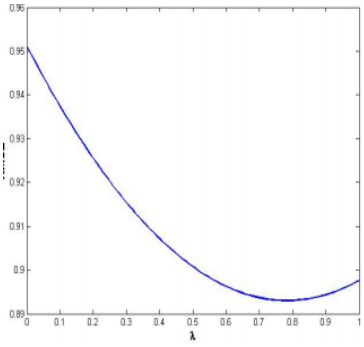

Fig. 7. Cross-validation for the model. Weight based on SVD.

There are important factors in the recommendation systems: user' interests, items' properties and other factors that directly affect user's preferences over items. Various kinds of information can be used to model these factors. For example, users browsing history over song reviews may correlate with user's taste over songs; the information of the artists and genre can be used to predict its properties; number of the times play song over similar song directly affect whether a user will favor the current one. Our model summarizes the three factors as feature vectors.

$$\hat{y}(\alpha,\beta,\gamma) = \left(\sum_{j=1}^{s} \gamma_j b_j^{(g)} + \sum_{j=1}^{n} \alpha_j b_j^{(u)} + \sum_{j=1}^{m} \beta_j b_j^{(i)}\right) + \left(\sum_{j=1}^{n} \alpha_j \mathbf{p}_j\right)^T \left(\sum_{j=1}^{m} \beta_j \mathbf{q}_j\right).$$

Fig.8. the model parameter set is defined as Θ = {b (g) ,b (u) ,b (i) ,p,q}. pj ∈R d and qj ∈R d are d dimensional latent factors associated with each feature. b (u) j , b (i) j and b (g) j are bias parameters

Let us suppose we want to recommend song for users using the playtime history, and we have known the artist and the playtime information. As we know, the album information can be used to better represent the content of tracks, and the temporal information can be used to detect the changes of item popularity. All this auxiliary information can help improve the performance of a recommender system. Taking them into consideration, we can represent a user u's preference towards song i at time t.

$$\hat{y}(u,i,t) = b_{i,bin(t)} + b_i + b_{p(i)} + b_u + \mathbf{p}_u^T(\mathbf{q}_i + \mathbf{q}_{al(i)})$$

Fig.9. Other auxiliary information can improve model.

D. Feature engineering

Good recommendation is optimization the problem. Since we are going to using Tensorflow, it is important to understand their feature engineering because it is matter of performance. It will make algorithms more effective with additional features. The feature engineering approach was the dominant approach till recently when matrix techniques started demonstrating recognition performance better than the carefully crafted feature detectors. Deep learning shifts the burden of feature design also to the underlying learning system along with classification learning typical of earlier multiple layer neural network learning. From this perspective, a SVD is a fully trainable system beginning from raw input, for example recommending music, to the final output of songs that user's preferred.

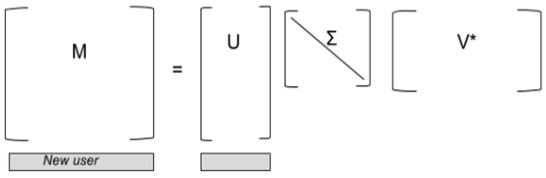

Fig.10. The schematics of the SVD folding-in technique

What we want to project this rating vector into our vector space, to do that we can utilize the decomposition above. Basically to add a new user, means to add another row to the rating matrix, which in turn would mean that the decomposed user matrix would also have another row, as illustrated figure 10.

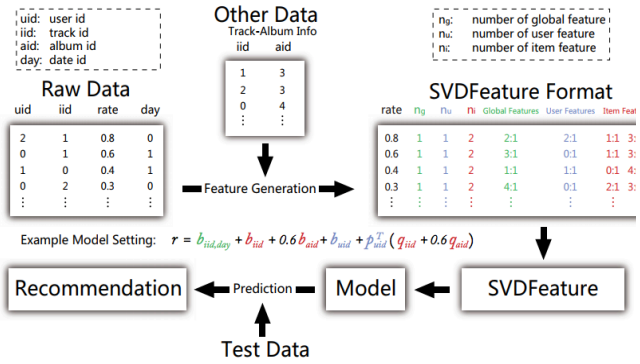

Fig.11. Flow chart for the SVD model.

Recommendation algorithms often have to deal with problems with large scale data set just like Rocksmith datasets, which has been taken into consideration in designing as feature engineering. In our approach, we store the data into a cloud service (google cloud). The data is shuffled before storing, and then the training program linearly iterates over the buffer and updates the model with each training sample. This approach allows us to do training as long as the model fits into. An independent thread is created to fetch the data from cloud into a memory queue. At the same time, the training thread reads the data from memory queue and updates the mode. This pipeline style of execution releases the burden of I/O from training thread.

Moreover, our model provides several other extra features for better modeling capability. It supports collaborative ranking training, different kinds of regularization options, and separated feature table extension for user and item features.

In the example, we assume to have 5,000 users and 1,000 songs. The number of factors(k) is rather high (500) can be dramatically reduced. The rating estimator are bounded between 1 and 5 (0 meaning no playing history). Every user has played 100 random songs and we want to suggest the top 5 songs. In this case, if the item has already been played, if fact these are situation (like Youtube) where the recommendations contain video frequently seen and other (like Netflix) where it's more useful to suggest new (never-seen) items.

Now, we have to think about who is never or very little played history on the system. Many of recommendation systems suggest them most popular product since we do not have enough evidence to recommend product. Since, we collected all the data from past year we easily create most popular song/artists that suits in this case. It will be significantly improving their activities just because recommend most popular items because once they started using this system it will be automatically personalized, and there is many business success story (Netflix, Youtube).

In our program once we build the user-song matrix we can calculate the recommendations for all users together with the matrix operations. Afterwards, indexing user ID we can get user's recommendations.

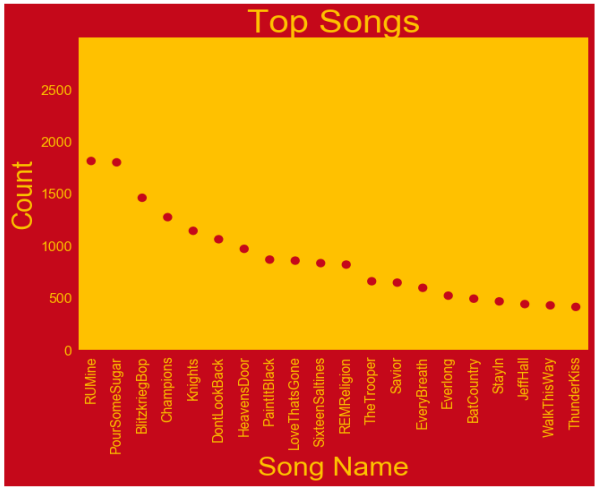

Fig.12. Most popular songs for new users (cold start)

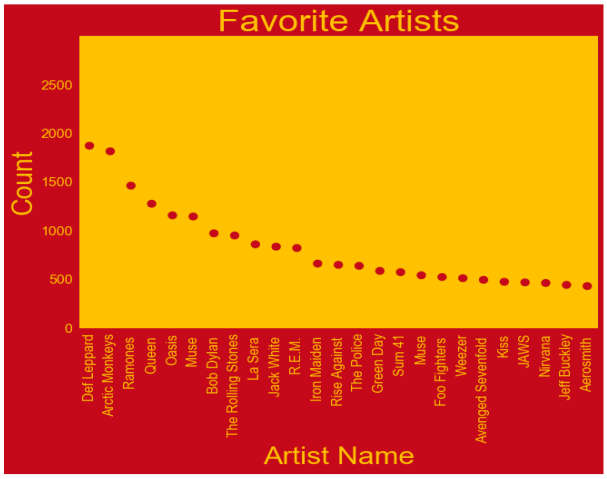

Fig.13. Most popular artists for new users (cold start)

Adding features and depth significantly improves precision on holdout data. In these experiments, over 11 Million playlists were embedded along with over 4K users.

IV. EXPERIMENTAL RESULTS

The overall, we finally found the optimal weight is $\lambda = 0.78$. With this optimal weight, the root-mean-square deviation (RMSD) for test data is 0.8930. In this project, we present collaborative filtering algorithms for recommendation system and test the performance of algorithm and their mixture on part of the Rocksmith data. At last a RMSE of 0.8930 is achieved, which corresponds to a 6.14% improvement of the performance of baseline.

We also observed that the most important signals are those that describe a user's previous interaction with the item itself and other similar items, matching other's experience in ranking. Features describing the frequency of past playlists are also critical for introducing "churn" in recommendations. If a user was recently recommended a video but did not watch it then the model will naturally demote this impression on the next time.

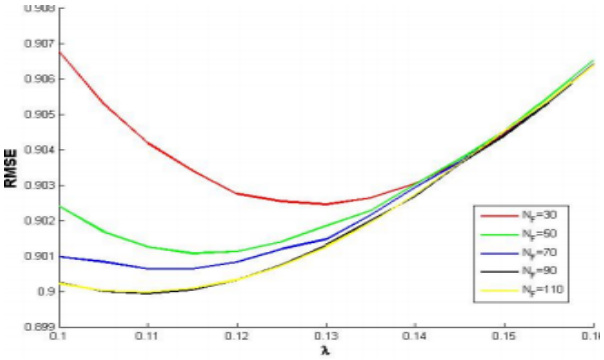

Fig.14. Cross validation for optimal weight.

After determined optimal weight we initiate to run whole 20GB data in cloud. In this model, input will be user ID and output will be top 5 songs most match to the users. For new users, they will recommend 5 most popular songs. This model will dynamically change the result in event such as user don't utilize this recommendation.

V. Conclusion And Future Research

A. CONCLUSION

It is safe to surmise that this project provides a tried-and-tested method, supported by credible results, for implementing Matrix Factorization to enhance the application of the Ubisoft product Rocksmith. Ultimately, this allows users to individualize their playlists. The decision to approach this dilemma from a song recommendation perspective was informed by the fact that it goes beyond gaming by offering a music playing functionality. SVD functioned optimally when developing matrix factorization because of our access to lots of data which was useful in analyzing numerical scores since they were absent in the data sets.

The disparity spawned by separating data is even more intense. User habits could be monitored by song start and stop in numerous ways, and the results imply that we can construct a model capable of taming fake subscribers and significantly improving users' ability to shuffle between playlists and songs. Other differences exist in the identification of the most popular songs and genres, which can remedy the problem associated with cold-start subscribers. There is sufficient evidence to validate the effectiveness of the recommendation technique on business and generate more innovative models in the future.

B. FUTURE RESEARCH

The criticality of the SVD model attributes reveals that each subscriber has unique patterns when using Rocksmith. Pitifully, the range of data available is presently limited to playlists. Personal data geography, actual reviews of song features, and choosing favorite genre before commencing the game can explicitly heighten the anticipation power. More specifically, in future similar projects should be rooted in new rather than active subscribers because we want to boost the commercial power of the product. Additionally, we are looking to try various algorithms, one of which is Bayes network. We are potentially combining a variety of algorithms to determine the most effective model.

REFERENCES

[1] Michael D. Ekstrand, John T Riedl and Joseph A. Konstan. Collaborative Filtering Recommender Systems.

[2] Guillaume Allain. Recommender systems with Tensorflow.

[3] Virginia C. Klema, Alan J. Laub. The Singular Value Decomposition: Its computation and Some applications.

[4] Giuseppe Bonaccorso, SVD Recommendations using Tensorflow

[5] Research Paper Recommender System Evaluation: A Quantitative Literature Survey.

[6] Stanford University. "Recommendation systems" Chapter 9.

[7] Zheng Wen. Recommendation System Based on Collaborative Filtering.

[8] Tianqi Chen, Weinan Zhang. SVDFeature: A Toolkit for Feature-based Collaborative Filtering.

[9] JeanFrancoisPuget. Feature Engineering For Deep Learning.

[10] Paul Covington, Jay Adams. Deep Neural Networks for YouTube Recommendation.