Comparing Deep learning techniques with traditional methods for Movie Recommendation System

**Ahmad Kamran, Kar Pratyusha, Patel Akshaykumar**

**CSYE 7245 Big Data Systems and Intelligence Analytics**

**Prof. Nik Bear Brown**

***Abstract*** — **In recent years, various commercials platforms are taking help of recommendation systems to provide better recommendations of items to users. Traditional recommendation algorithms like content-based and collaborative filtering algorithms are one of the main algorithms used in recommendation systems. These algorithms are simple and efficient but paying price in overall performance due to problems like sparsity in data, cold start, and scalability of method. Deep learning theory provides various methods which can solve these problems by mining the auxiliary data of user’s behavior and deeply mine the latent information of user features along with their correlations with item features. In this paper, deep learning models like Restricted Boltzmann machine, Deep Autoencoder, and simple deep neural network are used to overcome the sparsity and cold start problem efficiently. After that, the rating process is conducted to evaluate deep learning models and compared with traditional recommendation algorithms like latent schematic models (SVD and SVD ++) and a hybrid model with a combination of content-based filtering and SVD methods. Finally, after model comparison with Mean absolute error (MAE), it is verified that recommendation performance of Restricted Boltzmann machine is best among other models tried in this paper with MAE score of 0.044.**

# **INTRODUCTION**

**W**ith the increase in research for artificial intelligence technologies, more effective products are applied to solve various real-life problems and provides comforts to people in various aspects. The recommendation systems are example of one such intelligence products available in market which use deep learning technologies and help various platform to recommend various items to its user to increase their presence in the market for respective domains. Such recommendation systems are already used in various domain like e-commerce, movie and music platform

The recommendation algorithms are the part of recommender system which tries to analyze the massive data from the platform which contains the information regarding the user’s behavior and item available on that platform. These algorithms are nothings but various techniques which help us to process the data and give output based on the nature of the techniques. The output from these techniques are used to predict the behavior of the user towards the various item which is not been accessed by the same user. The commonly used algorithms which are used in most recommender system are content-based methods and collaborative filtering methods. In content-based methods, extra information of users and items like document content, user profiles, item attributes are used to make do recommendation. Mostly, the information which is available for user or item are difficult to obtain or even fake so this add limitation to content-based methods as recommendation system algorithm. Collaborative filtering methods are widely used as recommendation system algorithm and they do not require extra information of user and items and make good recommendation compared to content-based methods. Although these methods are simple and effective, but these methods have their own issue like sparsity of data and cold start due to which improvement was needed for these traditional methods.

Deep learning has achieved success in various domain such as computer vision, speech processing and image processing; however deep learning theory can be used for recommendation system which can help recommender system to overcome problems which are available in the traditional methods. In this paper, three different deep learning techniques are used to do movie recommendation on movie-lens dataset. The models are: Restricted Boltzmann machine, Deep Autoencoder, and simple deep neural network. These models have different architecture and methods to analyze the input data to overcome the problem of sparsity in the data. To evaluate these models, traditional algorithms like latent schematic models (SVD and SVD++) and hybrid model compared with the deep learning models. Results shows that deep learning model like Restricted Boltzmann machine have better results compared to other models which effectively increase the recommendation accuracy.

# **Methods**

The Following topics are related to various recommendation system algorithm information which are used to achieve our goal for this paper.

***A.* *Restricted Boltzmann machine***

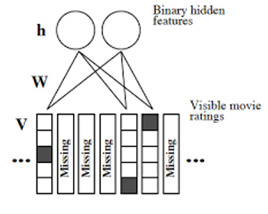
The structure of conventional Boltzmann Machine is such that each neuron in the network will be connected to every other neuron. This gives us a very complex structure and could reach deadlock when dealing with large size datasets. Whereas, the Restricted Boltzmann Machine comes with one restriction that no neuron in the visible and hidden layer would be connected to the neurons in the same layer. Thus, Restricted Boltzmann Machine is a two-layer undirected graph where no two hidden neurons or visible neurons are connected amongst each other.

A probability p(v, h) is assigned to each pair of a hidden and a visible vector:

p(v, h) = e −E(v,h) /Z

where E is the energy of the system and Z is a normalizing factor, as defined in (Hinton, 2002). To train for the weights, a Contrastive Divergence method was proposed by Hinton (Hinton, 2002). Salakhutdinov et al. (Salakhutdinov et al., 2007), proposed an RBM framework for CF. The model assumes one RBM for each user and takes only rated items into consideration when learning the weights.

Suppose we have dataset containing M (# of movies), N (# of users), and Rating of movies (values from 1 to K). The first problem in applying RBM’s to movie ratings is how to deal efficiently with the sparse data since each user in the dataset would have rated only very few movies. If we have a very sparse dataset, we utilize an alternate RBM for every user. Every RBM has the same number of hidden units, but an RBM only has visible softmax units for the movies rated by that user, so an RBM has only few connections if that user rated few movies only. The binary states of the hidden units (activate or not), however, can be quite different for different users.



*Figure 1*: This figure represents visible layer(V) which accepts movie rating data as input and hidden layer(h) which learns latent feature based on the input from visible layer(V).

## **B. Simple Deep Learning Neural Network**

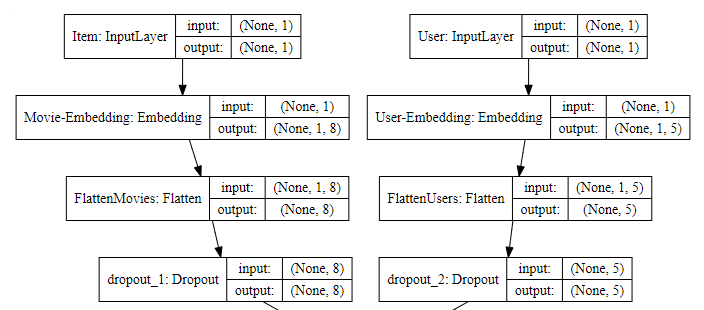
In this subsection, we will introduce the simple deep neural network which is consisting of two parts. First, model takes user and item data separately and perform user and item embedding. After that, Merge both matrixes after extracting latent features from user and item embedding which will feed to neural network. Second, neural network will take input by combining the user and item embedding matrices after flattening. Input passes to various neural network layers along with dropout layer to avoid over-fitting. At the output layer, neural network provides predicted ratings for a user which has not rated movie yet. This predicted ratings for that user will be used to recommend the new set of movies for a user.

According to our model, in the input layer of neural network takes two input vectors; user and item embedding metrices after flattering.



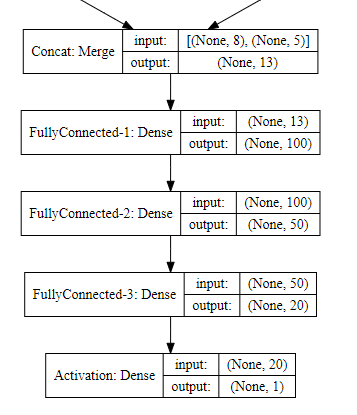
Where Ui = user embedding matrix

Vj = Movie embedding matrix



*Figure 2*: User and movie embedding matrices

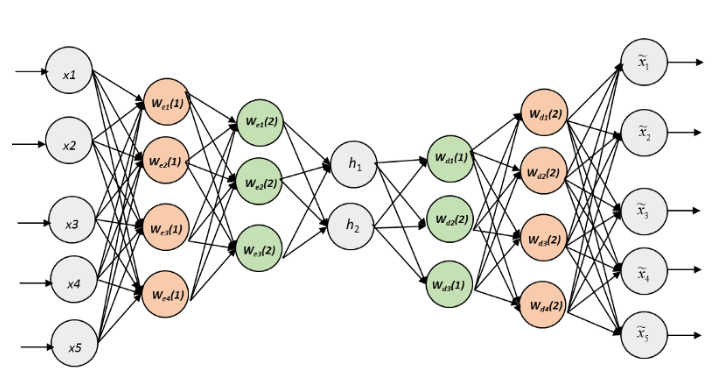
After merging of latent features matrices of user and item, x0 feeds to neural network to predict the user rating for the corresponding item.



*Figure 3*: Merge user and item embedding matrices and fed to neural network with 3 fully connected layer and one output layer.

## **C. Deep Autoencoder**

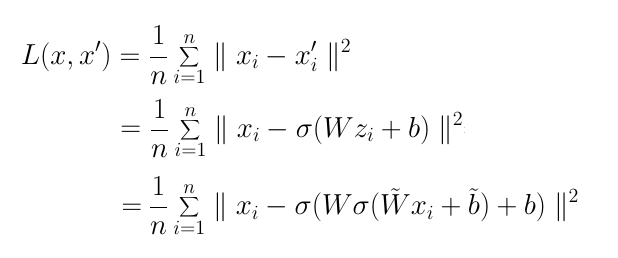
Deep Autoencoder is an important architecture in the deep learning model. The deep auto models used user-item matrices as input. The core purpose of the model is to reconstruct the output which is feed to input layer which means that the size of input layer in the autoencoder will be same as the size of output layer in the network. This solves the problem of non-integer scoring value prediction, but it does not add noise to input data which leads the algorithms less robust and more inclined towards over-fitting. During training period of autoencoder, it takes movie rating values for each user where the size of the input layer equals to number of items. It compresses input data from input layer till hidden layer. There might be multiple compression layer between input layer and hidden layer depending on the number of items available in the dataset. After compression, model will recreate the data from input by trying to minimizing the loss. Decompression of data starts from hidden layer till output layer. Output layer gives similar output as input layer after loss minimization by neural network which is predicted rating value for each item for a user.



*Figure 4*: Deep Autoencoder with input, hidden and output layers. Input and output layers have same size. Each node will multiples with random-normal weight.

Each input node value multiple with random normal weight along with constant bias and fed to activation function to create output for next layer. This method applies to each layer to compress and decompress the data in the network.

Stochastic gradient descent can be used to minimize loss such as mean squared error.



Where x = actual input, x’ = predicted output and L = loss

## **D. SVD and SVD++**

Matrix factorization is a class of collaborative filtering algorithms used in recommender systems. This algorithm works by decomposing the user-item interaction matrix into the inner product of two lower dimensionality rectangular matrices. The idea behind matrix factorization is to represent users and items in a lower dimensional latent space. One of the popular matrix factorization methods is Singular Value Decomposition (SVD). Singular Value Decomposition:

This algorithm decomposes the matrix into three small matrices, whereby we can obtain the features of users and items. SVD can be expressed as follows:

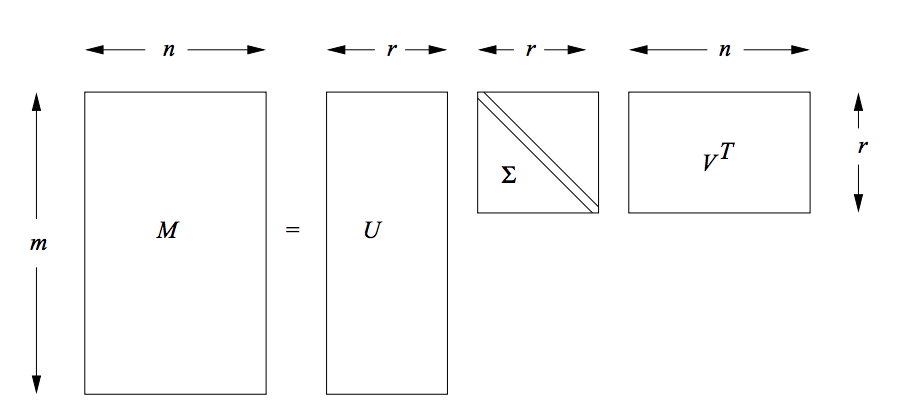
R = U · S · V t

Where R = Original matrix

U = User-Latent factor matrix

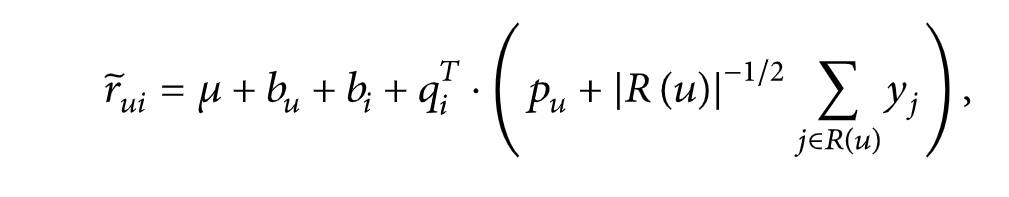
V = Item-Latent factor matrix (Here item is movie)

S = Diagonal Matrix (and the diagonal elements contain all the latent features). The predictive accuracy can be improved by considering the user and item bias information.



*Figure 5:* The SVD decomposition of an n × d matrix

The principal idea of SVD++ is to examine the user’s preference for each factor and the degree to which the movie contains the various factors from the observed ratings and some implicit feedback from users and then to predict the missing score. It introduces a factor vector () for each item, and these item factors are used to explain the characteristics of the item, regardless of whether it has been evaluated. Then, the user’s factor matrix is modelled, so that a better user bias can be obtained. In SVD++ the important hyperparameters (number of factors, number of epochs, learning rate and regularization term) is tuned to get a better evaluation factor.



r˜ui – each observed rating

µ - overall average rating

bu- user bias

bi- item bias

qi – item-factor matrix

pu -user-factor matrix

yi – Factor vector for each item

where R(u) is the number of items rated by user u.

## **E. Hybrid Model (Content-based method and SVD)**

Context Based Filtering is one of the common approaches of designing recommender systems is content-based filtering. Content-based filtering methods are based on the item and a profile of the user’s preferences. These methods are best apt to situations where there is known data on an item, but not on the user. Content-based recommenders treat recommendation as a user-specific classification problem and learn a classifier for the user's likes and dislikes based on items features. A content-based recommender works with data provided by the user, either explicitly or implicitly. Based on that data, a user profile is created, which is then used to make recommendation to the user. As the user provides more inputs or takes actions on the recommendations, the engine becomes more and more accurate.

*TF-IDF:*

The concepts of Term Frequency (TF) and Inverse Document Frequency (IDF) are used in content-based filtering mechanisms (such as a content-based recommender). They are used to determine the relative importance of a document / article / news item / movie etc. TF- IDF stands for Term Frequency and Inverse Document Frequency .TF-IDF helps in evaluating importance of a word in a document.

- TFN = (Number of times term t appears in a document) / (Total number of terms in the document), where n represents normalized.

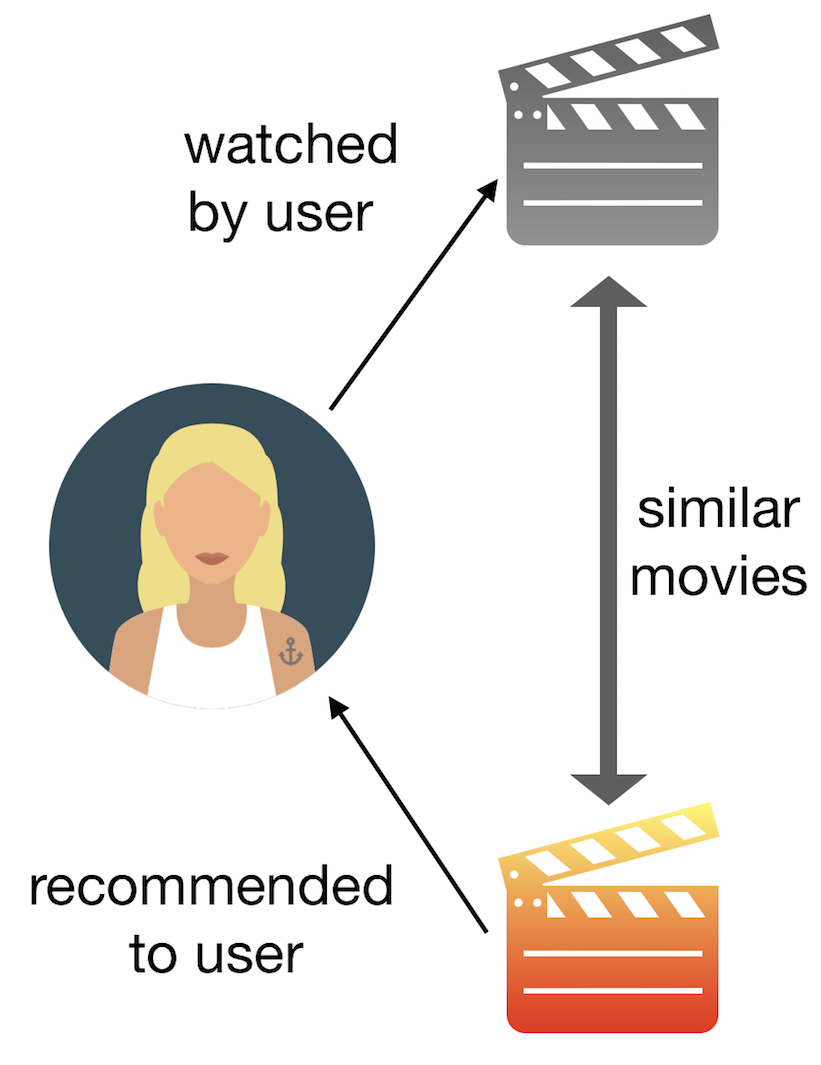
- IDF = ln (Total number of documents / Number of documents with term t in it)

Then TF-IDF weight is represented as:

TF-IDF Weight = TF (t,d) \* IDF(t,D)

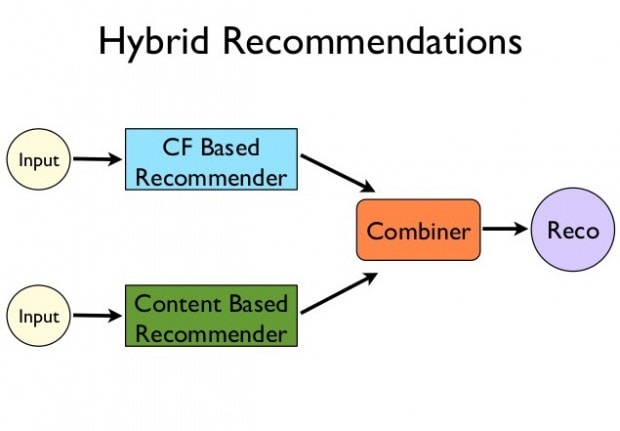
*Cosine Similarity:*

cosine similarity is a measure of similarity between two non-zero vectors. Through cosine similarity we can check how closely related two sentences are based on what angles their respective vectors make. Cosine value ranges from -1 to 1. So, if two vectors make an angle 0, then cosine value would be 1, which in turn would imply the sentences are closely related to each other. If the two vectors are orthogonal, i.e. cos 90 then it will imply the sentences are almost unrelated.



*Figure 6:* Content-based recommendation for a user

Hybrid matrix factorization algorithms can consider explicit and implicit interactions. This can also be modelled by using both content and collaborative data. Hybrid methodologies can be executed in several ways: by making content-based and collaborative-based predictions separately and then combining them; by adding content-based capabilities to a collaborative-based approach (and the other way around); or by binding together the methodologies into one model.

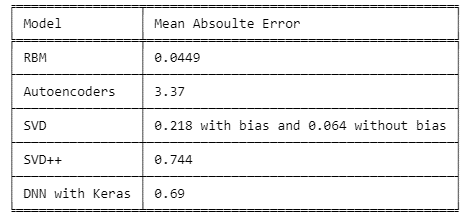


*Figure 7:*  Hybrid model based on collaborative filtering and content-based model.

# **Results**

After carefully fine tuning of our models we have evaluated our models based on MAE score; In deep learning models, **Restricted Boltzmann Machine** got MAE of **0.049**, **Autoencoders** got MAE of **3.37** and **DNN with Keras** got MAE of **0.69**.

In traditional model based collaborative approaches, **SVD** got MAE of **0.28** with bias and **0.064** without bias and **SVD++** got MAE of **0.744**.



*Figure 7:* MAE table for each model implemented in this paper.

# **Discussion**

We have implemented Restricted Boltzmann Machine, Deep Autoencoder, and simple deep neural network for this project to analyze which model performs better when implemented over large and sparse tabular dataset (in our case MovieLens-1m) and concluded that RBM performs slightly better than other deep learning and traditional models.

In **RBM**, the input contains X neurons, where X is the number of movies in our dataset. Taking hidden Units = 20, error\_function = MAE, epochs = 15 and activation function = {tf.sigmoid and tf.relu }, the model showed **MAE of 0.0449**. Deep Autoencoder algorithms used to predict the rating for user who has not rated movies. **Deep Autoencoder** algorithm predicted movie ratings with epochs=100, batch size= 100 and **MAE of 3.37** for test and 3.05 for train phase. **Simple deep neural network** takes user and movies as separate matrix and create embeddings on both matrices after that feature creation has done by taking dot product of the user and the item embeddings to feed in the network. Network has trained to predict the user rating for movie with **MAE of 0.69**. Using traditional algorithms, we attempted to build a model-based Collaborative Filtering movie recommendation system based on latent features from a low rank matrix factorization method called SVD and SVD++.For **SVD** model we get an **MAE of 0.218 with bias and 0.064 without bias** and for **SVD++** model we get an **MAE of 0.744**.

# **Scope**

The domain of the research covered most our prominent recommender models used in the industry currently. This research project focuses on performance analysis of various recommender system algorithms when dealing with very large sparse dataset (Movielens-1m). Analysis is carried out on various deep learning (such as Autoencoders, Restricted Boltzmann Machines, Deep Neural Networks) and traditional recommender system models (such as SVD’s, SVD++ and Hybrid) to determine how do each individual model performs when fed with large sparse dataset.

We analyzed each recommender system architecture in depth and then carefully tuned each one of them individually to determine how their performance vary with loss function, batch size, number of epochs, linear and non-linear activation functions. The study also included matrix factorization, contrastive divergence, WALS method for matrix factorization using TensorFlow and rating based on recommendation score when evaluating model performance.

Implemented techniques to overcome challenges related to recommender systems like Sparsity, Scalability, Latency problems, Cold start problems.

# **Context**

For deep learning methods we have referred a notebook (cited in the citation) where RBM model is implemented which was tweaked in our model by doing hyperparameter tuning, different network architecture and data normalization techniques to improve RBM’s performance.

For simple deep learning model, we have referred a research paper (cited in citation) which proposed simple deep learning model along with matrix factorization which uses user and item embedding matrices and creates features using dot product of both Matrix. In our model, user and item embedding matrices are merged to created input latent features for the neural network. Also, neural network optimized after tuning the hyperparameter to achieve best results for the dataset.

For deep autoencoder, we have tried to implement simplest version of deep autoencoder which presented in one of the researched papers (cited in citation). Our implementation have different architecture which we have experienced after rigorous tuning of hyperparameter tuning and network changes like number of compression of decompression layers.

We followed a notebook (cited in the citation) and changed matrix factorization technique implemented by using TensorFlow library. But we have implemented a different way of model evaluation. We have computed the MAE (Mean Absolute Error) by taking the difference between original user-item matrix and the reconstructed user-item matrix. We have also calculated the bias (globally, per user, per item, per item per user) and removed the bias to examine the model performances which have not been implemented in the referred notebook.

# **References**

1. https://en.wikipedia.org/wiki/Recommender\_system
2. https://towardsdatascience.com/creating-a-hybrid-content-collaborative-movie-recommender-using-deep-learning-cc8b431618af
3. https://en.wikipedia.org/wiki/Singular\_value\_decomposition
4. https://ieeexplore.ieee.org/Xplore/cookiedetectresponse.jsp
5. https://hpi.de/fileadmin/user\_upload/fachgebiete/naumann/lehre/SS2011/Collaborative\_Filtering/pres1-matrixfactorization.pdf
6. https://grouplens.org/datasets/movielens/1m/
7. https://en.wikipedia.org/wiki/Matrix\_factorization\_(recommender\_systems)
8. https://github.com/pratyushakar/factorizer/blob/master/factorizer.py
9. https://www.hindawi.com/journals/mpe/2017/1975719/
10. https://surprise.readthedocs.io/en/stable/matrix\_factorization.html#unbiased-note
11. https://towardsdatascience.com/recommender-engine-under-the-hood-7869d5eab072
12. https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8668497
13. https://towardsdatascience.com/deep-autoencoders-for-collaborative-filtering-6cf8d25bbf1d
14. https://arxiv.org/pdf/1708.01715.pdf
15. https://medium.com/@connectwithghosh/recommender-system-on-the-movielens-using-an-autoencoder-using-tensorflow-in-python-f13d3e8d600d
16. https://pdfs.semanticscholar.org/edca/6b95d76f9ec1c93ac6f334cb8153d5ebaf83.pdf
17. https://courses.cognitiveclass.ai/courses/course-v1:CognitiveClass+ML0120ENv2+2018/courseware/76d637cbe8024e509dc445df847e6c3a/d2529e01786a412fb009daef4b002a48/
18. https://medium.com/@tanaykarmarkar/explainable-restricted-boltzmann-machine-for-collaborative-filtering-6f011035352d
19. https://www.cs.toronto.edu/~rsalakhu/papers/rbmcf.pdf
20. https://chronicles.mfglabs.com/rbm-and-recommender-systems-3fa30f53d1dc
21. https://adityashrm21.github.io/Book-Recommender-System-RBM/
22. https://towardsdatascience.com/recommender-engine-under-the-hood-7869d5eab072