GALGOTIAS UNIVERSITY

(Established under Galgotias University Uttar Pradesh Act No. 14 of 2011)

PRODUCT BASED RECOMMENDATION SYSTEM ON AMAZON DATASET

A Project Report of Capstone Project – 2

Submitted by

ROHIT DWIVEDI

(16SCSE105134)

in partial fulfillment for the award of the degree

of

Bachelor of Technology

IN

Computer Science and Engineering with Specialization of Cloud Computing and Virtualization

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING

Under the Supervision of

DR. PRASHANT JOHRI, SCSE

Professor

APRIL - 2020

SCHOOL OF COMPUTING AND SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

Certified that this project report "**PRODUCT BASED RECOMMENDATION SYSTEM ON AMAZON DATASET**" is the bonafide work of "ROHIT DWIVEDI (1613105093) who carried out the project work under my supervision.

SIGNATURE OF HEAD

Dr. MUNISH SHABARWAL
PhD (Management), PhD (CS)
Professor & Dean,
School of computing science & Engineering

SIGNATURE OF SUPERVISOR

DR. PRASHANT JOHRI,
Professor
School of computing science & Engineering

TABLE OF CONTENTS

ABSTRACT

The hike of e-commerce has not only increased the choices but has also made information overloaded. In a way through which users can quickly find a favorite item from big resources, the user requires such technology by which it can be automated so recommendation systems were introduced. Recommendation systems are mainly used by companies like e-commerce to help the user discover items they have to found out by them and increase the sales of the company. The recommendation system can recommend the most relevant products to the customer. In this paper, we are introducing two recommendation system which uses a popularity-based recommendation system and a system which is build using collaborative filtering and SVD. These two recommendation systems are made over amazon's electronics dataset. Models are evaluated using RMSE and MAE also future advancements are discussed.

INTRODUCTION

There is a very large number of products that are listed today on e-commerce websites like Flipkart, Amazon, etc. Due to these many products often, users get confused about what to buy what not to buy. There are almost more than 30 million products present on Flipkart today. Due to which it has become tough for customers to choose their desired choice. So, to solve this problem recommendation system came into action which did magic behind the scenes by which it became easy for customers to select product of their interest. The recommender system deals with a large number of information present by filtering the most important information based on historical data of a user which takes care of the user's preference and interest. It can predict whether a user would prefer a product or not based on the user's historical data. These systems have proved to be beneficial to both the users as well as the service providers. The quality and decision-making process has also improved through these kinds of systems. Recommender systems result in mainly things stated below:

- Benefits users in finding items of their interest.
- Help item providers deliver their items to the right user.
- Identity products that are most relevant to the user.
- Personalized content.
- Help website improves user engagement.

An example of recommendation system has been discussed below:

There are two books which are listed on Amazon. The first book is ‘Touching the Void’ which was written by a mountaineer in 1988. At that time Amazon started as online book seller. But unfortunately, the book did not get much attention from the people and as a result, there were not many sales for that book. After a few years in 1996, another book came up with the title ‘Into thin Air’ which had similar story and got popular and received huge sales. That was the time when Amazon started with off implementing recommender system. So, whosoever bought ‘Into thin Air’ book, Amazon started recommending them ‘Touching the void’ book, as a result, the sales of the old book surpassed the new book. This magic happened because of recommendation. There are many such cases which proves recommender engine a good technology.

EXISTING SYSTEM

1.1 POPULARITY BASED RECOMMENDER SYSYEM

Let us take an example of a website that streams movies. The website is in its nascent stage and has listed all the movies for the users to search and watch. What the website misses here is a recommendation system. This results in users browsing through a long list of movies, with no suggestions about what to watch. This, in turn, reduces the propensity of a user to engage with the website and use its services. Therefore, the simplest way to fix this issue is to use a popularity-based recommendation system. Top review websites like IMDb and Rotten Tomatoes maintain a database of movies and their popularity in terms of reviews and ratings. Utilizing this data to recommend the most popular movies to users based on their star ratings, could increase their content consumption.
The popularity-based recommendation system eliminates the need for knowing other factors like user browsing history, user preferences, the star cast of the movie, genre, and other factors. Hence, the single-most factor considered is the star rating to generate a scalable recommendation system. This increases the chances of user engagement as compared to when there was no recommendation system.

1.2 CONTENT-BASED RECOMMENDER SYSTEM

This is content-based recommendation. Users or items have profiles describing their characteristics and the system would recommend an item to a user if the two profiles match. Stitch Fix’s fashion box is another example of content-based

recommendation. A user's attributes are collected (height, weight, etc.) and matched fashion products are put in a box delivered to the user (Stitch Fix|2013).

For Pandora, manual efforts/costs are needed to create music attributes, but there are many cases without such a need. Stitch Fix's customers provide their own preferences, LinkedIn users provide their own working experiences and skills, merchants on Amazon provide information about their product items, all are freely usable for content-based recommendations.

A straightforward way of matching users and items is keyword matching. For example, for job recommendations, one can match a job description to job seekers' resumes. Term frequency-inverse document frequency is often used to put more weights to keywords that are unique for an item or user.

A more systematic way is building a supervised model estimating the propensity of a user likes an unseen item. In the model, features are the attributes from users and items (e.g., an indicator variable whether a job and a job seeker is in the same industry), and the response variable is whether the user likes the item (e.g., whether the job seeker would apply for the job).

Content-based methods are computationally fast and interpretable. They can be easily adapted to new items or new users. However, some characteristics of items/users may not be easy to capture or describe explicitly. Stitch Fix addressed this by letting machine learning handle structured data, and human handle unstructured data (e.g., users' Pinterest board).

1.3 COLLABORATIVE FILTERING

Collaborative filtering systems make recommendations based on historic users' preference for items (clicked, watched, purchased, liked, rated, etc.). The preference can be presented as a user-item matrix. Here is an example of a matrix describing the preference of 4 users on 5 items, where p_{12} is the user 1's preference on item 2.

$$P = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} & p_{15} \\ p_{21} & p_{22} & p_{23} & p_{24} & p_{25} \\ p_{31} & p_{32} & p_{33} & p_{34} & p_{35} \\ p_{41} & p_{42} & p_{43} & p_{44} & p_{45} \end{bmatrix}$$

Although the entries can be numeric, e.g., Netflix's movie rating prediction challenge (rating ranges from 1 to 5), in most applications, they are binary (e.g., clicked, watched, purchased).

In reality, the user-item matrix can be more than millions * millions (e.g., Amazon, Youtube), and the majority of entries are missing — the goal of recommender systems is to fill those missing entries.

$$\begin{bmatrix} p_{11} & ? & p_{13} & ? & p_{15} \\ ? & ? & ? & p_{24} & ? \\ p_{31} & ? & p_{33} & ? & ? \\ ? & p_{42} & ? & p_{44} & p_{45} \end{bmatrix}$$

Here we describe three collaborative-filtering-related approaches, nearest-neighbor, and two methods creating a new latent space: matrix factorization and deep learning.

1.3.1 NEAREST NEIGHBOR

Nearest-neighbor-based methods are based on the similarity between pairs of items or users. Cosine similarity is often used for measuring the distance.

$$similarity(a,b) = cos(a,b) = \frac{a \cdot b}{||a|| * ||b||}$$

The preference matrix can be represented as item vectors

$$P = \begin{bmatrix} I_1 & I_2 & I_3 & I_4 & I_5 \end{bmatrix}$$

The similarity between item *I1* and item *I2* is calculated as *cos(I1,I2)*. The matrix can also be represented as user vectors

$$P = \begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \end{bmatrix}$$

The similarity between *U1* and *U2* is calculated as *cos(U1,U2)*. Note, the missing values in the preference matrix are commonly filled with zeros.

For user_i, we can recommend the items liked by user_i's most similar users (user-to-user) or the most similar items of user_i's liked items (item-to-item).

Item-to-item approaches are adapted commonly in practice, by Amazon (Amazon|2003), Youtube (Youtube|2010), Linedin (Linkedin|2014), etc. When a customer likes an item, an item-to-item system can quickly surface items similar to

it (similar items for each item are pre-calculated and saved in a key-value data store). In addition, item-to-item recommendations can be more interpretable than user-to-user recommendations, e.g., the systems can explain why an item is recommended because "*you liked X*".

It is possible that the number of items similar to an item is too small (after applying a threshold on the similarity scores). One could expand the similar item list by including similar items' similar items (Youtube|2010).

After getting the most similar items, a post-processing step can be useful. (Youtube|2010) ranked the similar items according to video quality (e.g., measured by rating), diversity (e.g., limited the recommendations from one channel), and user specificity (e.g., videos similar to a video a user watched more should be ranked higher for the user). The three elements were combined with a linear model, providing a final ranking.

1.4 LATENT-FACTOR METHODS

Latent-factor methods create a new and usually reduced feature space of the original user or item vectors, leading to reduced noise and faster computations in real-time.

In the following, we introduce two latent-factor methods — matrix factorization and deep learning.

1.4.1 MATRIX-FACTORIZATION

Matrix factorization was popularly used during the Netflix recommendation challenge, especially singular value decomposition and a more practical version for recommender systems.

Singular value decomposition *(*SVD) decomposes the preference matrix as

$$P_{m\times n} = U_{m\times m} \sum_{m\times n} V_{n\times n}$$

U and *V* are unitary matrices. For 4 users and 5 items, it looks like

$$P_{4\times 5} = \begin{bmatrix} u_{11} & u_{12} & u_{13} & u_{14} \\ u_{21} & u_{22} & u_{23} & u_{24} \\ u_{31} & u_{32} & u_{33} & u_{34} \\ u_{41} & u_{42} & u_{43} & u_{44} \end{bmatrix} \bullet \begin{bmatrix} \sigma_1 & 0 & 0 & 0 & 0 \\ 0 & \sigma_2 & 0 & 0 & 0 \\ 0 & 0 & \sigma_3 & 0 & 0 \\ 0 & 0 & 0 & \sigma_4 & 0 \end{bmatrix} \bullet \begin{bmatrix} v_{11} & v_{12} & v_{13} & v_{14} & v_{15} \\ v_{21} & v_{22} & v_{23} & v_{24} & v_{25} \\ v_{31} & v_{32} & v_{33} & v_{34} & v_{35} \\ v_{41} & v_{42} & v_{43} & v_{44} & v_{45} \\ v_{51} & v_{52} & v_{53} & v_{54} & v_{55} \end{bmatrix}$$

where *sigma_1 > sigma_2 > sigma_3 > sigma_4*.

The preference of the first user for the first item can be written as

$$p_{11} = \sigma_1 u_{11} v_{11} + \sigma_2 u_{12} v_{21} + \sigma_3 u_{13} v_{31} + \sigma_4 u_{14} v_{41}$$

This can be presented as vectors

$$p_{11} = \overrightarrow{\sigma} \circ \overrightarrow{u_1} \bullet \overrightarrow{v_1}$$

An entry wise product is applied between the sigma vector and the first user vector, and then a dot product with the first item vector. It can be seen *u* and *v* have the same

length, i.e., they are in the same latent feature space. The sigma vector represents the importance of each feature.

Now let's select the top two features based on the sigmas

$$p_{11} \approx \sigma_1 u_{11} v_{11} + \sigma_2 u_{12} v_{21}$$

which can be presented as the item and user vectors each has a length of two.

1.4.2 SVD

Lots of entries in the preference matrix can be missing and the regular SVD has the following issues (1) how missing values are imputed can have an undesirable impact on the outcome. (2) computational complexity for training can be high with all the entries considered.

During the Netflix challenge, Simon Funk came up with a practical solution (Funk|2006)

$$min_{\overrightarrow{u_i},\overrightarrow{v_j}} \sum_{p_{ij}} (p_{ij} - \overrightarrow{u_i} \cdot \overrightarrow{v_j})^2$$

In the formula, only the non-missing entries *p_{ij}* are considered. The estimated score for the *j^{th}* item from *i^{th}* user is

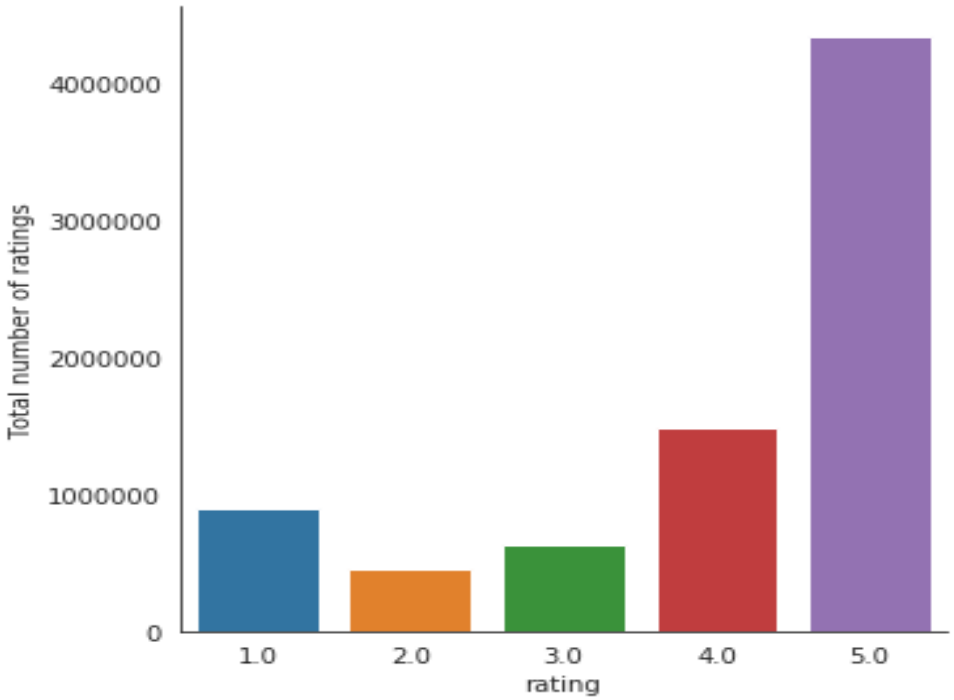
$$\overrightarrow{u_i} \cdot \overrightarrow{v_j}$$

Note, user and item vectors do not have unit lengths as in SVD, but it doesn't matter

as the sum of squares for error is minimized. Funk's approach had great success in the Netflix challenge, and the idea was implemented by Netflix (Netflix|2012).

PROPOSED SYSTEM

Before proceeding to the building of the model at first bit of data analysis was done so as to get familiar with the data. There was a total of 7824481 rows and 3 columns. Missing value checking was done and was found that there were no missing values. Plots was plotted for the rating respect to users and it was found ratings ranged from 1-5. The subset of the data was taken to make it less sparse and dense and only those users who have rated more than 50 products was taken in consideration. There were about 12587 users who have rated more than 50 products. The final data was divided into 70:30 training and test respectively. Data shape:(37762, Train data shape:(88109, 4).



X-axis represent ratings and Y-axis represent users.

MODEL – (I)

Popularity-based recommender system where we first calculated how many users have rated a particular product using python package for data manipulation which is pandas and NumPy. Stored the number of ratings corresponding to a respective product in a column and gave it index as ‘score’. The score described count of the rating users has rated to the particular product. Then it was grouped in descending order with respect to products and printed the most popular products which had highest ratings which were then recommended to every user. Prediction of the products was done based on popularity.

Here is the recommendation for the userId: 121

	user_id	prod_id	score	Rank
30847	121	B0088CJT4U	133	1.0
30287	121	B007WTAJTO	124	2.0
19647	121	B003ES5ZUU	122	3.0
8752	121	B000N99BBC	114	4.0
30555	121	B00829THK0	97	5.0

Here is the recommendation for the userId: 200

	user_id	prod_id	score	Rank
30847	200	B0088CJT4U	133	1.0
30287	200	B007WTAJTO	124	2.0
19647	200	B003ES5ZUU	122	3.0
8752	200	B000N99BBC	114	4.0
30555	200	B00829THK0	97	5.0

Recommendation based on system(I) for user 123 and 200.

MODEL – (II)

Using matrix-factorization build another model using a user-based nearest-neighbor collaborative filtering approach. Initially, using pivot function na pivot table was made based on the user which meant where ever the user has rated a product the respective ratings is present and those products where the user has not given the rating, it was filled with 0. So, matrix of user and product was obtained. Replaced the userID by creating a different column which was user index using np. arrange () function in NumPy giving step size of 1 and till the last row in the training data. So, finally, in the data frame, there were user from 0 to last row index of the training set, respective product, and respective rating. Used the SVD algorithm to train the training set where it was seen what each user has rated to each product. After, training our algorithm predicted each user rating with each product and compared it with the original rating. Created a function and recommend the product to users who had given similar ratings to the product. In short, the system was made in such a way that on the basis of similar ratings the system reverts the similar users and recommendation of the product is done. Calculated the top 5 recommendations using the same function i.e. (k=5). The system was found to bit personalized system.

FUTURE ENHANCEMENT

HYBRID RECOMMENDER SYSTEM

Multiple recommender systems are combined to improve recommendations Although any type of recommender systems can be combined a common approach in industry is to combine content-based approaches and collaborative filtering approaches.

Content based models can be used to solve the Cold Start and Gray Sheep problems in Collaborative Filtering.

Some of the typical methods of Hybridization include:

- Weighted –Recommendations from each system is weighted to calculate final recommendation.
- Switching –System switches between different recommendation model.
- Mixed - Recommendations from different recommenders are presented together. A common approach is to use Latent Factor models for high level recommendation and then improving them using content-based systems by using information on users or items.

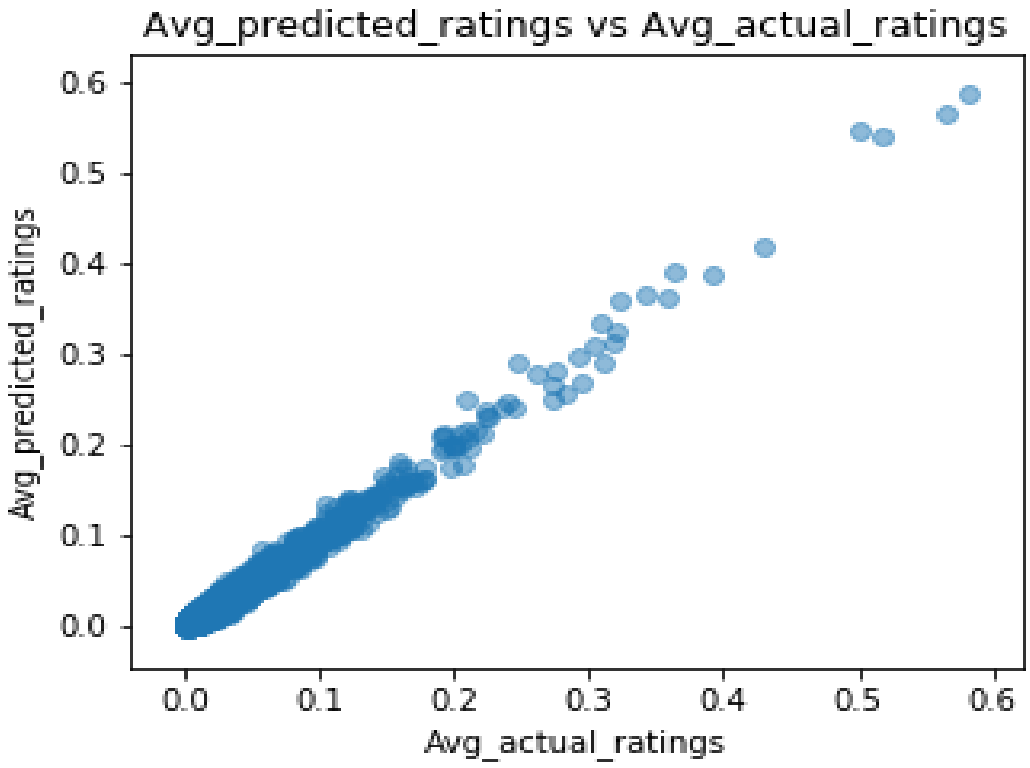
Deep Neural Network for recommendation systems:

YouTube recommendation system is based on Deep Learning, one of the largest and most complex recommendation systems.

RESULT

$$\mathrm{MAE} = \frac{1}{n}\sum_{j=1}^{n}|y_j - \hat{y}_j| = 0.00275$$

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(y_j - \hat{y}_j)^2} = 0.0019$$



Scatter plot between actual ratings and predicted rating.

CONCLUSION

Model-based Collaborative Filtering is a personalized recommender system, the recommendations are based on the past behavior of the user and it is not dependent on any additional information.

The popularity-based recommender system is non-personalized and the recommendations are based on frequency counts, which may be not suitable to the user. You can see the difference above for the user id 121 & 200 as shown in figure 3, The Popularity based model has recommended the same set of 5 products to both but the Collaborative Filtering based model has recommended entirely different list based on the user past purchase history. Deep neural networks can be used to improve the recommendations. YouTube used deep neural network architecture for doing recommendation.

REFERENCES

[1] https://medium.com/voice-tech-podcast/a-simple-way-to-explain-the-recommendation-engine-in-ai-d1a609f59d97https://www.geeksforgeeks.org/supervised-unsupervised-learning/

[2] .https://arxiv.org/ftp/arxiv/papers/1703/1703.09109.pdf

[3] https://www.scss.tcd.ie/joeran.beel/pubs/2016%20IJDL%20---%20Research%20Paper%20Recommender%20Systems%20--%20A%20Literature%20Survey%20(preprint).pdf

[4] https://dl.acm.org/doi/pdf/10.1145/245108.245121

[5] https://www.researchgate.net/publication/200121027_Collaborative_Filtering_Recommender_Systems

[6] http://files.grouplens.org/papers/FnT%20CF%20Recsys%20Survey.pdf

[7] https://www.hindawi.com/journals/aai/2009/42142/