Movie Recommendation System

Venkat Koushik Muthyapu

Computer Science, University Of New Mexico

ABSTRACT

Movies are an important part of how people can view and connect to current and past cultures. They can provide deep emotional messages, as well as bringing people together. And with more and more streaming services becoming available to the public it suddenly becomes much more important for these companies to properly recommend movies that they know their customers will enjoy. That is the purpose of this work; a movie recommendation system is important in our social lives, as well as for businesses due to its ability to recommend movies based on a user's interests, as well as the popularity of the movie. We took a dataset named MovieLens which had 100,000 ratings from 1000 users on 1700 movies and created a collaborative filtering movie recommender system. This paper presents the approach that we took and show the results of our final recommender program using the collaborative filtering method.

CCS Concepts

- Information Systems → Recommender Systems

Keywords

Recommender System: Content Based Filtering, Collaborative Filtering; ALS; MovieLens; Matrix Decomposition

1. INTRODUCTION

There is a continually growing reliance upon the internet in today's society as a result of which it has become a necessary part of human life. The main problem that people are facing now is that there are a lot of options to choose from. The internet has allowed connections to extend all the way around the world, so now people have more and more access to alternative products and/or items. In order to assist customers in making their choices companies have begun creating recommendation systems. This helps to guide their customers to the products that they are most likely to use based on certain information. Research into these systems has been going on for a while and many people are still interested because the number of fields where the recommendation systems can be used for is growing at a very rapid rate as a result of advancement in technology.

Recommender Systems are used to suggest items that might be of interest of the user based on their own preferences or based on the interest of a similar user. Based on this they are classified into three techniques:

- Content based Recommender system: This kind of recommendation system makes use of a user-specific classification problem and does the learning of a classifier based on user's likes and dislikes on product features. Thus, it only relies on the user-item interactions.

- Collaborative Filtering based Recommender system: This kind of recommender system makes use of historical preference of users on a set of items. The core assumption made by this technique here is that users who have agreed in the past tend to also agree in the future. The interactions of all the users are stored in the so-called "user-item interactions matrix".

- Hybrid Recommender System: This kind of recommender system makes use of a combination of the previous two approaches.

Normally recommendations are done by looking at the user profile and its previous behavior [11]. The most popular recommender systems currently in use are those of Amazon, YouTube, and Netflix. The systems help the users to find and select the items that they are most likely to enjoy (e.g., movies, books, videos, locations) from the collection that is available on the website, by implementing the recommender system. Among a large set of items and a description of the user's needs and preferences, they supply a small set of the items to the user that are well suited based on that information. That is also the purpose for the recommender system that we have designed. A movie recommendation system should provide a certain amount of comfort and personalization to the user, so they can better interact with the system and watch the movies that they are most likely to enjoy. The main

purpose of our system is to recommend movies based on the rating provided by the user and other people from the MovieLens survey dataset and personalize them for each user. To do so the first step that should be done is the collection of the data from the user. However, this not an easy step. To do this there are two techniques:

- Explicit
- Implicit

In the first technique, the user is asked to rate the movies on scale which is different for different platforms (such as rating a movie
from one to five stars). This is an easy technique if the users rate for all movies. Unfortunately, most of the users tend to not rate the movies and in such a scenario we need a different method.

In the first technique, the user is asked to rate the movies on scale which is different for different platforms (such as rating a movie from one to five stars).
This is an easy technique if the users rate for all movies. Unfortunately, most of the users tend to not rate the movies and in such a scenario we need a different method.

This is to gather data implicitly as the user is in the domain of the system - that is, to log the actions of a user on the site. This is much better since most users tend to not rate the movies explicitly. In this technique, a large amount of data can be easily gathered with no extra effort from the user. The drawback of this technique is that it's difficult to work on. In this case, the data of user in raw form must be made to a scale of preference by the system.

The two methods are not mutually exclusive with each other. To obtain the best way to collect the data we need a combination of both these techniques.

A. Content-based Recommendation

A content-based filtering system takes in information based on the interest of a single user based on his interests and history.

This methodology can be done by finding similarity between all pairs of items and then choosing an item similar to the users rated item to generate a list of recommendations. Usually the similarity is obtained from the description of the item. The similarity between the descriptions are computed using TF-IDF scores. This technique is used to count the concurrence of each word in a document and weight them according to their importance, and also calculate a score for the document. This same process can be applied to movies by applying a weight to different thematic elements and words used to describe movies, such as genres.

For use the method builds a user profile and item profile based on user rated content. The term frequency (TF) and the inverse document frequency (IDF) can be computed as shown

$$\mathrm{Tf}(t) = \frac{\text{Frequency occurence of term t in document}}{\text{Total number of terms in document}}$$

$$\mathrm{Idf}(t) = log_{10}\left(\frac{\text{Total Number of documents}}{\text{Number of documents containing term t}}\right)$$

$$\textbf{TF-IDF score: } w_{ij} = TF_{ij} \times IDF_i$$

Once the TF and IDF are found we find the TF-IDF score by taking a product of these two for each movie vector in the dataset. This gives a score for each item in our dataset. Now once we have the scores we need to compute the similarity between each item using cosine similarity. [10]

Given user profile $\boldsymbol{x}$ and item profile $\boldsymbol{i}$, estimate

$$u(\boldsymbol{x}, \boldsymbol{i}) = \cos(\boldsymbol{x}, \boldsymbol{i}) = \frac{\boldsymbol{x} \cdot \boldsymbol{i}}{||\boldsymbol{x}|| \cdot ||\boldsymbol{i}||}$$

This can provide recommendations based on most similar items.

However, there are certain drawbacks to this kind of recommender systems:

- Content based recommender system lacks novelty. It does not suggest diverse contents to the customer. In most cases, it can be observed that movies are being suggested just on the interest of the customer and lacks to show any new type of content.
- Content based recommender systems lacks scalability. In the world of big data, where there are lots of information, it requires scalability for better performance.

B. Collaborative Filtering

A collaborative Filtering system takes the information regarding how, not just a single user, but how other users too, have previously rated the movies

within the database. They then consider the user's preference for movies and base their recommendations on how other users with similar tastes rated other movies.

The collaborative filtering approach has two main approaches for handling the data. A memory based approach and a model based approach. For memory based collaborative filtering there are two main approaches, user-item filtering and item-item filtering. In the first approach the system finds similar users based on their similarity of ratings and recommends the items that are liked by both. In the latter approach the recommendations are done by taking an item and looking at the users that have liked it, and then finding other items that are also liked by those users.

The other approach to collaborative filtering is the model based method. In this approach, the system uses a machine learning algorithm in order to predict the ratings of unrated items. There are many different algorithms that can be used for this approach, but the most commonly used ones are clustering based, matrix factorization, and deep learning.

A clustering based algorithm is very like a memory based algorithm. Where the similarities are found using cosine similarity, but in this system the data is being calculated as an unsupervised learning model in order to improve the scalability of the function.

The next algorithm is the Matrix factorization based algorithm. In this technique, the data is represented as a matrix and then the matrix is broken down into two smaller matrices, such that when multiplied they give the original matrix. When the matrix is multiplied, a few parameters are used to predict the unrated products. [8]

And the last algorithm is the deep learning algorithm. This technique centers on neural networks that are used to implement the collaborative filtering. [9]

However, there are certain drawbacks to this kind of recommender systems too:

- Cold Start: This kind of recommender systems requires a certain number of users to rate the movie before the recommendations can work efficiently.
- Sparsity: The user/ratings matrix is sparse. Most of the users either tend to not rate the movies after watching or in other cases may not have watched the movies.
- It is difficult to find the users who have rated the same movies.
- Popularity bias: Since this recommender system works based on the interest of many users there will be cases where always the most popular movies would be recommended. As a result, system fails to recommend movies to someone with unique tastes.

Thus, we are effectively solving a data sparsity problem. To work on such a kind of problem we use an ML algorithm called matrix factorization which is efficiently able to solve the problem. To do this we have different algorithms and which will be described in the subsequent sections

2. RELATED WORK

The topic of recommender systems has been a very wide research topic. There are different approaches that have been tried in the past to solve this problem. In [1], the algorithm that was proposed to solve the problem was K-means and KNN algorithm and they performed a comparison with cuckoo search algorithm and were able to obtain better results. However, as the number of clusters increased RMSE increased and thus this approach was not viable since it lacks scalability. The minimum RMSE value achieved started from 1 which implied that the model was not particularly effective. In [2], they did a memory based CF using the algorithm SVD (Singular Value Decomposition). This is also a matrix decomposition technique. However, they used only a small amount of data to train the model and thus they were not able to achieve better RMSE values. But they did achieve better RMSE than other memory based collaborative filtering technique. In this paper [3], the main aim was to implement the balanced memory based collaborative filtering by combining two different approaches, Jaccard mean square differences and Jaccard Similarities. But after a few attempts they implemented only JMST since the combination of both could not provide the desired results. The dataset used had one million ratings. A content based approach was implemented in [4], where a recommender system was implemented for digital library to recommend research papers. However, the results here were too personalized to a particular user and devoid him a chance of exploring new topics. Thus, this system was not very effective but the authors were satisfied with the result as they tested it against an existing system. In [5], an ALS based recommender system was implemented for a particular movie recommender system. However, based on the parameters selected for implementations, it affected the performance. This was a major learning curve in our research since it really pointed out how choosing the parameters carefully affects the RMSE

value. The RMSE achieved for the particular experiment was around 0.9. However, in [6], the implementation was done using the same ALS algorithm, but by choosing the parameters carefully. On doing so they were able to achieve far better RMSE values of 0.8. This was the area of focus for our project and was inspiration for our work.

3. Development of Movie Recommendation System

This section explains the algorithms and code used in the recommendation system that we developed. Along with how we gathered the user's data to provide personal recommendations. And finally, an image of our testing prototype is displayed to provide a view of the finished product.

3.1 Final Decisions

After going through a great amount of research on recommender systems we have come to the conclusion that the matrix factorization algorithm would be the best reasonable algorithm based on the type of data that we have. Tor be precise, we are using the Alternative Least Square algorithm (ALS) because it is one of the best algorithms that can deal with sparse data, has high scalability along with the ability to predict with new data that has no background and efficient run time compared to other matrix factorization techniques.

ALS is an iterative optimization process where every iteration attempts to get closer and closer to a factorized representation of the original data. It starts with an original matrix R of size *u* x *i* with user, items, and some feedback. The goal is to make two matrices, one of size *f* x *i* with items and features, and another of size *u* x *f* with users and features. The new matrices U and V have weights for how each user/item relates to each feature. U and v are calculated so that their product approximates R as best as possible with randomly assigning the values and using least squares iterative to find what weights best approximate R.

Our rating matrix R consists of users as rows and movies as columns. This matrix is then factored into two smaller matrices P and Q. The P matrix consists of rows and columns where rows represent latent features and the columns represent moves. Whereas Q consists of rows and columns where rows represent user and the columns latent features. [7]

$$Error_{ij} = \underbrace{\sum w_{ij}}_{\text{Completion Term}} \cdot \underbrace{(R_{ij} - u_i \times p_j^T)}_{\text{Cost Function}} + \underbrace{\lambda(\| U \|_2 + \| P \|_2)}_{\text{Regularization Term}}$$

where $w_{ij} \begin{cases} 1 & R_{ij} \text{ is known} \\ 0 & R_{ij} \text{ is unknown} \end{cases}$

Completion Term

Cost Function
Minimizes the difference between the product of our factor matrices and the original ratings matrix.

Regularization Term
Prevents overfitting by applying a small amount to the error which requires more time/iterations to full minimize.

The matching solutions for u_i and p_j are:

$$u_i = (P^T \times w_i \times P + \lambda I)^{-1} \times P^T \times w_i \times r_i$$

$$p_j = (U^T \times w_j \times U + \lambda I)^{-1} \times U^T \times w_j \times r_j$$

In the Als model, we are trying to alternately minimize two cost functions (error), where the first error function holds the user matrix fixed and runs gradient descent with the movies matrix, whereas the second error function holds the movies matrix and runs a gradient descent with respect to the user matrix. This is much more scalable than other matrix decomposition techniques, such as PCA, because ALS is running the gradient descent algorithm over multiple partitions of training data from a cluster of machines.

We increase the number of latent factors to improve the personalization. But, after a certain point of time, the number of factors become too high and the model starts to over fit. To prevent this we introduced the regularization parameter to the above equation. Thus, the above equation on a whole minimized the error between true rating and predicted rating. [12]

3.2 Implementation

To make our recommendation system we have implemented an ALS machine learning algorithm which makes use of matrix decomposition. The MovieLens data set consists of four csv files, movies.csv, ratings.csv, links.csv and tags.csv. These csv files were loaded as tables and then reformatted into RDDs using pyspark. This was necessary since the data being worked on was very large and thus it was big data. Without using the concepts of big data, it is tedious to implement a Recommender System. Also as ALS algorithm has high scalability the use of big data made the process of execution faster. For the implementation of our recommender system we made use of movies and ratings table since we tried to predict the ratings for unrated movies based on user ratings.

The data in the ratings table was then split at random into three sets; a training dataset, which consisted of 60 percent of the data, a validation dataset, which consisted of 20 percent of the data, and finally the test data set, which consisted of the remaining 20 percent of data. The ALS model has three important parameters, maxIter, Rank and regParam. The maxIter is the maximum number of iterations to run. This was set to 10 in our model. The Rank is the number of latent features in the model, for our model we set the rank to 18. The regParam is the regularization parameter which was set to 0.05 in our model.

The parameters listed above are used to train our training data. After the training, we tested our model on sample test data and we can see that we have a RMSE (Root Mean Square Error).

```
For rank = 18 reg = 0.05  the RMSE=  0.9803222190248909
```

From the above results, it is evident that our model was very close to over fitting. To avoid this outcome, we have followed an evaluation method as follows.

4. EVALUATION

Evaluating effectiveness of recommendation system. For the purpose of finding a reasonable ALS model for our dataset we have come across some results in [6] after analyzing the outcome from those parameters and some amount of research we have decided to use the following values for the parameters. The maxIter was kept at 10. The rank was changed to help prevent overfitting. The model was tested with [8, 10, 12, 14, 16, 18, 20]. The regParam was tested with [0.001, 0.01, 0.05, 0.1, 0.2] for our model. After that, we iterated through with those parameters and trained the system using the training data set. Along with performing cross validation on the validation dataset. Through our testing, we obtained the following results to help us find the best model.

```
For rank = 8 reg = 0.001  the RMSE=  1.4467448634372828
For rank = 8 reg = 0.01  the RMSE=  1.141916955112322
For rank = 8 reg = 0.05  the RMSE=  0.9856996845615988
For rank = 8 reg = 0.1  the RMSE=  0.914186924444086
For rank = 8 reg = 0.2  the RMSE=  0.8966076281249638
For rank = 10 reg = 0.001  the RMSE=  1.4560396253085752
For rank = 10 reg = 0.01  the RMSE=  1.2036612472694024
For rank = 10 reg = 0.05  the RMSE=  0.9956121741381563
For rank = 10 reg = 0.1  the RMSE=  0.9158487323974703
For rank = 10 reg = 0.2  the RMSE=  0.8964889415116521
For rank = 12 reg = 0.001  the RMSE=  1.5242743892266395
For rank = 12 reg = 0.01  the RMSE=  1.235516337012286
For rank = 12 reg = 0.05  the RMSE=  1.0054946510104066
For rank = 12 reg = 0.1  the RMSE=  0.9178090013903913
For rank = 12 reg = 0.2  the RMSE=  0.8985851115100361
For rank = 14 reg = 0.001  the RMSE=  1.5311798036079438
For rank = 14 reg = 0.01  the RMSE=  1.2574262798813998
For rank = 14 reg = 0.05  the RMSE=  1.004432265748177
For rank = 14 reg = 0.1  the RMSE=  0.9163648847902194
For rank = 14 reg = 0.2  the RMSE=  0.899009456306896
For rank = 16 reg = 0.001  the RMSE=  1.5977755203925776
For rank = 16 reg = 0.01  the RMSE=  1.2945525589067843
For rank = 16 reg = 0.05  the RMSE=  1.004848074487897
For rank = 16 reg = 0.1  the RMSE=  0.9149908997725777
For rank = 16 reg = 0.2  the RMSE=  0.8988097775919106
For rank = 18 reg = 0.001  the RMSE=  1.64173527955219
For rank = 18 reg = 0.01  the RMSE=  1.3049537155223991
For rank = 18 reg = 0.05  the RMSE=  1.008261040265912
For rank = 18 reg = 0.1  the RMSE=  0.915377249956471
For rank = 18 reg = 0.2  the RMSE=  0.8988576731676956
For rank = 20 reg = 0.001  the RMSE=  1.7042923866197208
For rank = 20 reg = 0.01  the RMSE=  1.3538113042962867
For rank = 20 reg = 0.05  the RMSE=  1.0154965096385316
For rank = 20 reg = 0.1  the RMSE=  0.9160800093109257
For rank = 20 reg = 0.2  the RMSE=  0.8999188491754373
The best model was trained with rank=  10 With reg=  0.2
```

Table 1: This table represents the RMSE errors for different values of the aforementioned parameters and we find that best model is obtained at rank=10 and reg= 0.2

From the above evaluations, the best model was found to be the one with rank = 10 and regparam = 0.2. This model gave an RMSE value of 0.896488. Comparing to the implementations in the related works we were able to achieve an RMSE value similar to that of [6]. Furthermore, this confirmed our initial belief on the ALS algorithm compared to other algorithms implemented in recommender systems

4.1 Analysis of Data

The table shown below provides the comparisons between the user ratings and the ratings predicted by our model joined with userId and movieID displaying the top 20 results.

userId	movieId	rating	prediction
1	2143	4.0	3.4857223
1	2959	5.0	4.8959446
1	736	3.0	3.546459
1	2078	5.0	4.5469894
1	3273	5.0	2.521188
1	3527	4.0	4.1616406
1	733	4.0	4.186975
1	2987	5.0	4.081792
1	2141	5.0	3.6555433
1	2654	5.0	3.6944335
1	423	3.0	3.2964041
1	553	5.0	4.2281413
1	1025	5.0	4.2760987
1	1136	5.0	4.7466288
1	47	5.0	4.542998
1	1270	5.0	4.675041
1	367	4.0	3.7773545
1	1	4.0	4.5714073
1	500	3.0	3.9910703
1	1226	5.0	3.6889038

only showing top 20 rows

Table 2: This table shows the actual ratings and predicted ratings for user 1 on running the algorithm on test data.

Looking at the values on the table indicate that our predicted values are somewhat close to the actual ratings for those particular movies. There are some ratings that a clearly closer than others but we feel confident enough in our code that we feel our system is still a viable option. We also created histograms showing the distributions of ratings by users and the predicted ratings in the dataset, as can be seen below. From the histograms, it's evident that the mean of the data is almost the same in both the cases. However, it is observed that the standard deviation has been reduced. This is because in the implementation, the number of iterations was set to only 10 and as we increase it, the distribution starts to look more similar. However, it takes more time to predict in such a case and it's a tradeoff between accuracy and time required. When the same implementation is run for more iterations it gives an RMSE value around 0.85. Thus, more the iteration more the accuracy. Also, one other observation made to improve the result was to increase the rank. As the rank is increased the model gives us a better RMSE value. But, this should be done under extremely careful since if the ranks is increased beyond a certain value the model tends to over fit. This should be avoided in order to have recommender system with no bias.

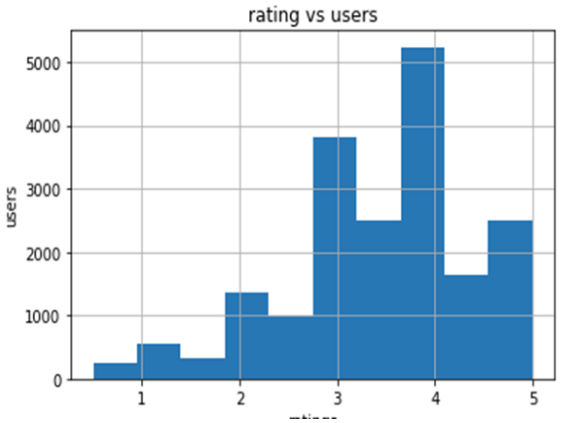

Figure 1: The figure represents a histogram of actual ratings plotted against the users. The mean of this plot is 3.515 and standard deviation of the plot is 1.035.

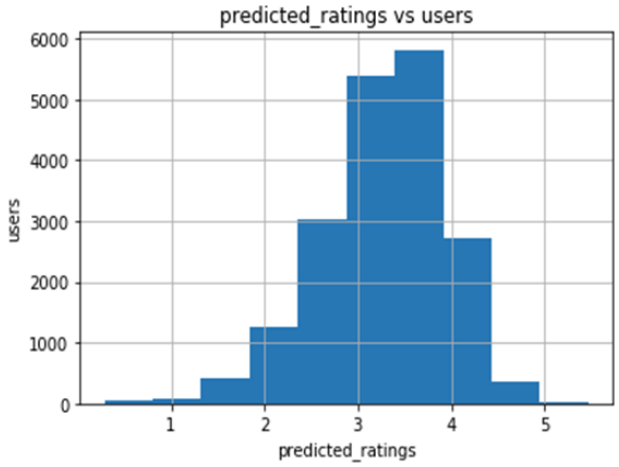

Figure 2: The figure represents a histogram of predicted ratings plotted against the users. The mean of this plot is 3.272 and standard deviation of the plot is 0. 669.

4.2 Final model of Recommender System

The best model that was obtained by evaluation on the validation data set was then tested on the test data, which was split earlier, but never used, to validate the model. This gives a RMSE value as below:

```
test data Rmse=  0.893580776686068
```

From the above RMSE on the test data set it became evident that the model had an RMSE less than that on the validating data, which, to an extent, shows that it does not over fit.

5. CONCLUSION

Based on our research on recommender systems throughout the project, and through our own testing we feel confident in stating that the implementation of the collaborative filtering system using ALS was successful. Through this process we were also able to learn more about the alternating least squares algorithm and the model. It gave insight onto how the way the parameters are chosen results affects the recommender systems. Furthermore certain key observations from the implementation is as below:

- As the number of the maxIter increased the value of the RMSE decreased which allowed us to find a more consistent value that prevented overfitting.
- Regularization is required on the model to prevent overfitting, and finally that when the regularization parameter is around 0.2, we observed even smaller RMSE values. We have tried to implement the ALS learning curve but was unable to due to certain dimensionality issues.

6. FURTHER DIRECTIONS

The next steps that we would take in the implementation of our recommender system would be attempting to create a hybrid recommender system. This type of system takes the better parts of a content based recommendation system and a collaborative filtering recommendation system and combines them together to hopefully provide even better recommendations. Along with implementing the full dataset, which we could not do in the time that we had because of how long the system took to run if we included more data.

Overall our recommender system was able to provide mostly accurate results when recommending movies based on the systems and algorithms that we have implemented.

7. CONTRIBUTION STATEMENT

The implementation of the project was done by each member of the group and the best implementation was chosen.

8. REFERENCES

[1] Movie Recommender System Using K-Means Clustering AND K-Nearest Neighbor Available at: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8776969 [2] Grover, P. (2019). *Various Implementations of Collaborative Filtering*. [online] Towards Data Science. Available at: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6455211

[3] A balanced memory-based collaborative filtering similarity measure: https://onlinelibrary.wiley.com/doi/pdf/10.1002/int.21556
[4] Application of Content-Based Approach in Research Paper Recommendation System for a Digital Library: https://pdfs.semanticscholar.org/c9f9/6d22422953625f1f8d9dbee221cba38e6c08.pdf
[5] Movie Recommender System Based on Collaborative Filtering Using Apache Spark Available at: https://www.researchgate.net/publication/327536169_Movie_Recommender_System_Based_on_Collaborative_Filtering_Using_Apache_Spark_Proceedings_of_ICDMAI_2018_Volume_2
[6] COMPARATIVE ANALYSIS OF MOVIE RECOMMENDATION SYSTEM USING COLLABORATIVE FILTERING IN SPARK ENGINE Available at: https://www.researchgate.net/publication/321058803_COMPARATIVE_ANALYSIS_OF_MOVIE_RECOMMENDATION_SYSTEM_USING_COLLABORATIVE_FILTERING_IN_SPARK_ENGINE
[7] Liao, K. (2018). *Prototyping a Recommender System Step by Step Part 2: Alternating Least Square (ALS) Matrix Factorization in Collaborative Filtering*. [online] Towards Data Science. Available at: https://towardsdatascience.com/prototyping-a-recommender-system-step-by-step-part-2-alternating-least-square-als-matrix-4a76c58714a1 [Accessed 13 Dec. 2019].
[8] Liao, K. (2018). *Prototyping a Recommender System Step by Step Part 1: KNN Item-Based Collaborative Filtering*. [online] Towards Data Science. Available at: https://towardsdatascience.com/prototyping-a-recommender-system-step-by-step-part-1-knn-item-based-collaborative-filtering-637969614ea [Accessed 13 Dec. 2019].
[9] Lineberry, A. and Longo, C. (2018). Creating a Hybrid Content-Collaborative Movie Recommender Using Deep Learning. [online] Towards Data Science. Available at: https://towardsdatascience.com/creating-a-hybrid-content-collaborative-movie-recommender-using-deep-learning-cc8b431618af [Accessed 13 Dec. 2019].
[10] Luk, K. (2019). *Introduction to TWO approaches of Content-based Recommendation System*. [online] Towards Data Science. Available at: https://towardsdatascience.com/introduction-to-two-approaches-of-content-based-recommendation-system-fc797460c18c [Accessed 13 Dec. 2019].
[11] Mahowald, M. (2019). *Building a Recommender System*. [online] Kd nuggets. Available at: https://www.kdnuggets.com/2019/04/building-recommender-system.html [Accessed 13 Dec. 2019].
[12] Poudyal, R. (2018). *Latent Factor based method in collaborative filtering*. [online] Medium. Available at: https://medium.com/@rabinpoudyal1995/latent-factor-based-method-in-collaborative-filtering-77756a02f675 [Accessed 13 Dec. 2019].