IJCSI International Journal of Computer Science Issues, Volume 13, Issue 5, September 2016
ISSN (Print): 1694-0814 | ISSN (Online): 1694-0784
www.IJCSI.org https://doi.org/10.20943/01201605.110

Recommender System in Big Data Environment

Udeh Tochukwu Livinus[1], Rachid Chelouah[2] and Houcine Senoussi[3]

Quartz Laboratory, EISTI,
Avenue du Parc, Cergy, 95000, France

Abstract

Recommender systems are an Artificial Intelligence technology that has become an essential part of business for many industries and businesses. Recommender Systems (RSs) are software tools and techniques providing suggestions for items to be of use to a user. The system learns from a customer and recommends products that she will find most valuable from among the available products. They serve many types of E-commerce applications, from direct product recommendation for an individual to helping someone find a gift for a third party [10]. Recently the world of the web has become more social and more real-time. Facebook and Twitter [16] are perhaps the exemplars of a new generation of social, real-time web services and we believe these types of service provide a fertile ground for recommender systems research. In this research project, the researcher tries to provide a present an analysis of how recommender systems can be used in E-commerce today, companies, SMEs, and other social institutions to improve sales, maintain good customer relationship and loyalty, and saves customers sourcing time. Recommender Systems too can also be used to analyze Big Data, although there are some key challenges associated with Big Data analytics. The impact of data abundance extends well beyond business [15]. But the computer tools for gleaning knowledge and insights from the Internet era's vast trove of unstructured data are fast gaining ground. At the forefront are the rapidly advancing techniques of artificial intelligence like natural-language processing, pattern recognition and machine learning. The researcher hopes that these key problems with improved recommender systems in the future: hybrid data, predictable recommendations, scalability, and incorporation of content, such problems could be resolved. If recommender systems are able to surmount these challenges, they have the potential to become an essential component of doing business in Ecommerce

Keywords: *Recommender System, SparkR, Collaborative Filtering, K-Means, KNN, Content Based Method, Clustering Hadoop, MapReduce, Million Song Data*.

1. Introduction

Recommender systems or recommendation systems are a subclass of information filtering system that seek to predict the 'rating' or 'preference' that a user would give to an item. Recommender systems have become extremely common in recent years, and are applied in a variety of applications. The most popular ones are probably movies, music, news, books, research articles, search queries, social tags, and products in general. However, there are also recommender systems for experts, jokes, restaurants, financial services, life insurance, persons (online dating), and Twitter followers. The first Recommender System is widely recognized in the literature as Tapestry (Goldberg, Nichols, Oki, & Terry, 1992) which was an experimental mail system developed at the Xerox Research Centre in Palo Alto over 20 years ago. From this starting point, the literature discusses collaborative and content-based filtering methods, with some modifications, for example with statistical elements also included. Rather than providing a static experience in which users search for and potentially buy products, recommender systems increase interaction to provide a richer experience.

Recommender systems identify recommendations autonomously for individual users based on past purchases and searches, and on other users' behavior. Examples of such systems are a system based on music data grouping and user interests (Hung-Chen & Chen, 2014) which utilizes three approaches – Collaborative, Content-based and Statistical – to predict recommendations for users. Another system based on Deep Content is proposed by Van den Oord et al (Van Den Oord, Dieleman, & Schrauwen, 2013) which utilizes deep convolutional neural networks and compares results to a more traditional "Bag-of-words" approach. Jayalakshmi et al (Jayalakshmi, Shruthi, Sneha, & Uttarika Ratnakar, 2014) combine collaborative and content-based filtering for their Hybrid Music Recommender System which they found performed better than either of the combined systems when used alone.

This paper will use a similar approach in that there is a combination of collaborative filtering and content-based (MSD attributes) input to the system. However, the user input is unique in that it takes the user's attributes into consideration in the collaborative filtering stage and we will try to explore the performance of item based collaborative filtering and user based collaborative filtering.

1.1 Related work

In this section we briefly present some of the research literature related to collaborative filtering, recommender systems, data mining and personalization. Tapestry is one of the earliest implementations of collaborative filtering-based recommender systems. This system relied on the explicit opinions of people from a close-knit community, such as an office workgroup. However, recommender system for large communities cannot depend on each person knowing the others. Later, several ratings-based automated recommender systems were developed. The GroupLens research system [3, 4] provides a pseudonymous collaborative filtering solution for Usenet news and movies. Ringo [5] and Video Recommender [6] are email and web-based systems that generate recommendations on music and movies respectively. A special issue of Communications of the ACM presents a number of different recommender systems other technologies have also been applied to recommender systems, including Bayesian networks, clustering, and Horting. Bayesian networks create a model based on a training set with a decision tree at each node and edges representing user information. The model can be built off-line over a matter of hours or days. The resulting model is very small, very fast, and essentially as accurate as nearest neighbor methods [7]. Bayesian networks [8] may prove practical for environments in which knowledge of user preferences changes slowly with respect to the time needed to build the model but are not suitable for environments in which user preference models must be updated rapidly or frequently.

Clustering techniques work by identifying groups of users who appear to have similar preferences. Once the clusters are created, predictions for an individual can be made by averaging the opinions of the other users in that cluster. Some clustering techniques represent each user with partial participation in several clusters. The prediction is then an average across the clusters, weighted by degree of participation. Clustering techniques usually produce less-personal recommendations than other methods, and in some cases, the clusters have worse accuracy than nearest neighbor algorithms [7]. Once the clustering is complete, however, performance can be very good, since the size of the group that must be analyzed is much smaller. Clustering techniques can also be applied as a "first step" for" for shrinking the candidate set in a nearest neighbor algorithm or for distributing nearest-neighbor computation across several recommender engines. While dividing the population into clusters may hurt the accuracy or recommendations to users near the fringes of their assigned cluster, pre-clustering may be a worthwhile trade-off between accuracy and throughput. Horting is a graph-based technique in which nodes are users, and edges between nodes indicate degree of similarity [9] between two users. Predictions are produced by walking the graph to nearby nodes and combining the opinions of the nearby users. Horting differs from nearest neighbor as the graph may be walked through other users who have not rated the item in question, thus exploring transitive relationships that nearest neighbor algorithms do not consider. In one study using synthetic data, Horting produced better predictions than a nearest neighbor algorithm. Schafer et al., present a detailed taxonomy and examples of recommender systems used in E-commerce and how they can provide one-to-one personalization and at the same can capture customer loyalty [10]. Although these systems have been successful in the past, their widespread use has exposed some of their limitations such as the problems of sparsity in the data set, problems associated with high dimensionality and so on. Sparsity problem in recommender system has been addressed in. The problems associated with high dimensionality in recommender systems have been discussed in, and application of dimensionality reduction techniques to address these issues has been invest. However, with the discovery of data mining tools and knowledge [11] which inherently is associated with databases more robust approaches can be used to analyze large datasets and make recommendation more quick and easy.

Our work explores the extent to which item-based recommenders, a new class of recommender algorithms, are able to solve these problems.

1.2 Project description

This project is designed in five sections. The first section describes the essence of the project and Introduction. Section two gives us an introduction of works about Recommender Systems with examples. Section three of this project is the Art state, followed by the data analysis of the researcher. The next section is the result of findings from the researcher and the last section of this project is the conclusion.

The final part is the references as regards to the project conducted in similar domain. The Dataset for this study was extracted from Amazon database [1].

2 State of art

A preprocessed dataset was downloaded from Amazon database which is in hdf5. In order to read the hdf5 files in which the songs and their attributes were stored it was necessary to download some software such as WinPython from Sourceforge (Dice Holdings Inc., 2014). WinPython is a portable Python distribution which allows building self-contained C extensions. If you want to deploy your CPyMAD build to other machines, this is the

distribution of choice. The software provides the necessary libraries for importing the data into MySQL and to access the hdf5 files via hdf5 'getters'.

The libraries required were Numpy, Math and Pytables which were needed to operate the hdf5 getters code, which was downloaded from Github (GitHub Inc., 2014). The Python libraries for connecting to MySQL were in Numpy.

The dataset is a large file so we took a subset of the data in order to make our analysis.

Algorithms implemented

A typical way to evaluate a prediction is to compute the deviation of the prediction from the true or actual value. This is the basis for the Mean Average Error (MAE)

$$\mathrm{MAE} = \frac{1}{|\mathcal{K}|} \sum_{(i,j)\in\mathcal{K}} |r_{ij} - \hat{r}_{ij})|, \quad (1)$$

Where K is the set of all user-item pairings (i, j) for which we have a predicted rating $\hat{r}_{ij}$ and a known rating r_{ij} which was not used to learn the recommendation model. Another popular measure is the Root Mean Square Error (RMSE).

$$\mathrm{RMSE} = \sqrt{\frac{\sum_{(i,j)\in\mathcal{K}}(r_{ij} - \hat{r}_{ij})^2}{|\mathcal{K}|}} \quad (2)$$

RMSE penalizes larger errors stronger than MAE and thus is suitable for situations where small prediction errors are not very important.

2.1 Evaluation Top-n recommendations

The items in the predicted top-N lists and the withheld items liked by the user (typically determined by a simple threshold on the actual rating) for all test users Utest can be aggregated into a so called confusion matrix depicted in table 2 (see Kohavi and Provost (1998)) which corresponds exactly to the outcomes of a classical statistical experiment. The confusion matrix shows how many of the items recommended in the top-N lists (column predicted positive; d +b) were withheld items and thus correct recommendations (cell d) and how many where potentially incorrect (cell b).

The matrix also shows how many of the not recommended items (column predicted negative; a + c) should have actually been recommended since they represent withheld items (cell c). From the confusion matrix several performance measures can be derived. For the data mining task of a recommender system the performance of an algorithm depends on its ability to learn significant patterns in the data set. Performance measures used to evaluate these algorithms have their root in machine learning. A commonly used measure is accuracy, the fraction of correct recommendations to total possible recommendations.

Table 1: Confusion Matrix

Actual/Predicted	Negative	Positive
Negative	a	b
Positive	c	d

The following statistical computation can be analyzed from the table above: Accuracy which is the systematic errors as follows:

$$Accuracy = \frac{correct\ recommendations}{total\ possible\ recommendations} = \frac{a+d}{a+b+c+d} \quad (3)$$

$$\text{accuracy} = \frac{\text{number of true positives} + \text{number of true negatives}}{\text{number of true positives} + \text{false positives} + \text{false negatives} + \text{true negatives}} \quad (4)$$

An accuracy of 100% means that the measured values are exactly the same as the given values. A common error measure is the mean absolute error (MAE, also called mean absolute deviation or MAD) is also computed as follows:

$$MAE = \frac{1}{N}\sum_{i=1}^{N} |\epsilon_i| = \frac{b+c}{a+b+c+d}, \quad (5)$$

Where N = a+b+c+d is the total number of items which can be recommended and |εi| is the absolute error of each item. Since we deal with 0-1 data, |εi| can only be zero (in cells a and d in the confusion matrix) or one (in cells b and c). For evaluation recommender algorithms for rating data, the root mean square error is often used. For 0-1 data it reduces to the square root of MAE. Recommender systems help to find items of interest from the set of all available items. This can be seen as a retrieval task known from information retrieval. Therefore, standard information retrieval performance measures are frequently used to evaluate recommender performance. Precision and recall are the best known measures used in information retrieval (Salton and McGill 1983; van Rijsbergen 1979). Precision or positive predictive value is defined as the proportion of the true positives against all the positive results (both true positives and false positives)

$$\text{precision} = \frac{\text{number of true positives}}{\text{number of true positives} + \text{false positives}} \quad (6)$$

$$Precision = \frac{correctly\ recommended\ items}{total\ recommended\ items} = \frac{d}{b+d} \quad (7)$$

$$Recall = \frac{correctly\ recommended\ items}{total\ useful\ recommendations} = \frac{d}{c+d} \quad (8)$$

Another method used in the project to compare two classifiers at different parameter settings is the Receiver Operating Characteristic (ROC). The method was developed for signal detection and goes back to the Swets model (van Rijsbergen 1979). The ROC-curve is a plot of the system's probability of detection (also called sensitivity or true positive rate TPR which is equivalent to recall as defined in formula (13) by the probability of false alarm (also called false positive rate FPR or 1−specificity, where specificity = a / (a+b)) with regard to model parameters. A possible way to compare the efficiency of two systems is by comparing the size of the area under the ROC-curve, where a bigger area indicates better performance. *Sensitivity and specificity* are statistical measures of the performance of a binary classification test, also known in statistics as classification function:

- Sensitivity (also called the true positive rate, or the recall in some fields) measures the proportion of positives which are correctly identified as such (e.g., the percentage of sick people who are correctly identified as having the condition), and is complementary to the false negative rate.
- Specificity (also called the true negative rate) measures the proportion of negatives which are correctly identified as such (e.g., the percentage of healthy people who are correctly identified as not having the condition), and is complementary to the false positive rate.

For any test, there is usually a trade-off between the measures. These two statistical measures can be computed as follows:

$$\text{sensitivity} = \frac{\text{number of true positives}}{\text{number of true positives} + \text{number of false negatives}} \quad (9)$$

$$TPR = TP/P = TP/(TP + FN) \quad (10)$$

$$\text{specificity} = \frac{\text{number of true negatives}}{\text{number of true negatives} + \text{number of false positives}} \quad (11)$$

$$SPC = TN/N = TN/(TN + FP)$$

2.2 Recommenderlab infrastructure

Recommenderlab is implemented using formal classes in the S4 class system. The Figure below shows the main classes and their relationships. The package uses the abstract ratingMatrix to provide a common interface for rating data. RatingMatrix implements many methods typically available for matrix-like objects. For example, *dim*(), *dimnames*(), *colCounts*(), *rowCounts*(), *colMeans*(), *rowMeans*(), *colSums*() and *rowSums*(). Additionally *sample*() can be used to sample from users (rows) and *image*() produces an image plot.

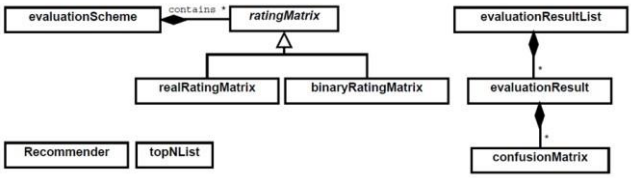

Figure 1: Recommendation Algorithm Architecture

Often the number of total useful recommendations needed for recall is unknown since the whole collection would have to be inspected. However, instead of the actual total useful recommendations often the total number of known useful recommendations is used. Precision and recall are conflicting properties, high precision means low recall and vice versa. To find an optimal trade-off between precision and recall a single-valued measure like the E-measure (van Rijsbergen 1979) can be used. The parameter α controls the trade-off between precision and recall.

$$E\text{-}measure = \frac{1}{\alpha(1/Precision) + (1-\alpha)(1/Recall)} \quad (12)$$

Popular single-valued measure is the F-measure. It is defined as the harmonic mean of precision and recall.

$$F\text{-}measure = \frac{2\ Precision\ Recall}{Precision + Recall} = \frac{2}{1/Precision + 1/Recall} \quad (13)$$

It is a special case of the E-measure with α = .5 which places the same weight on both, precision and recall. In the recommender evaluation literature the F-measure is often referred to as the measure F1.

Recommendation System Model

In order to develop our model, we need the following packages:

- recommenderlab - Create a list or data frame representation for various objects used in recommenderlab. These functions are used in addition to available coercion to allow for parameters like decode.
- matrix – creates matrix from the given data frame.
- registry – used for matching functions to be specified for key fields

- arules – item-matrix is defined in this package. This object is created by the call of "binaryRatingMatrix" and defining the data as 'im' which means item-matrix.
- Proxy – used for distance and similarity measures. Provides an extensible framework for the efficient calculation of auto and cross – proximities.
- ggplot2 – used for creating elegant and complex plots.

3. Contributions

With coercion, the matrix can be easily converted into a realRatingMatrix object which stores the data in sparse format. Here, you can see that our matrix is very large. That is 47402 X 42900 rating matrix of class 'realRatingMatrix' with 47402 ratings.

An important operation for rating matrices is to normalize the entries too, e.g., remove rating bias by subtracting the row mean from all ratings in the row. This can be easily done using normalize () function.

We can represent the raw matrix and the normalized matrix with the image syntax.

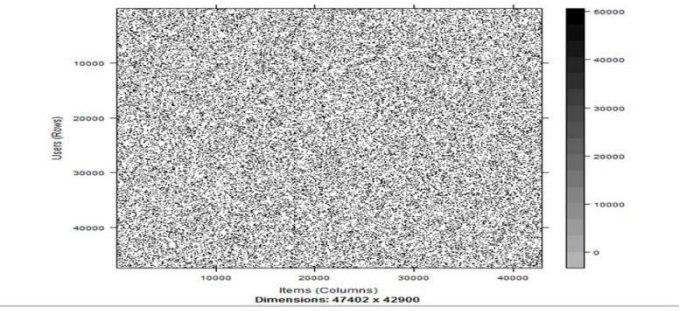

Figure 2: Raw ratings

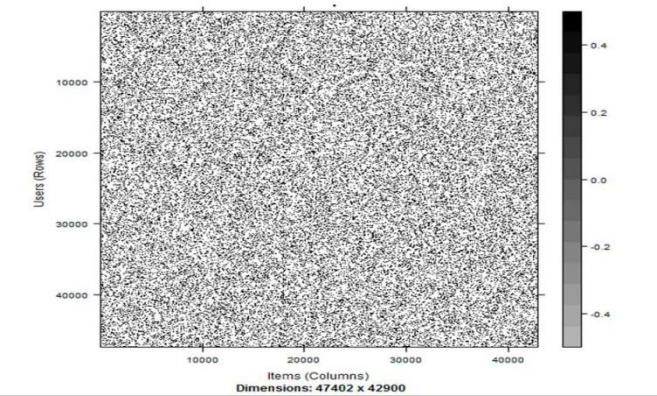

Figure 3: Normalized ratings

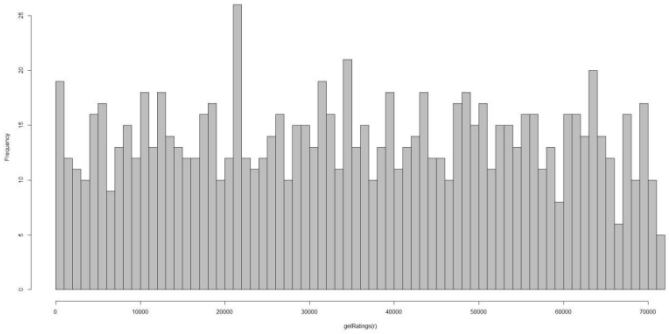

Figure 4: Histogram of getRatings(r)

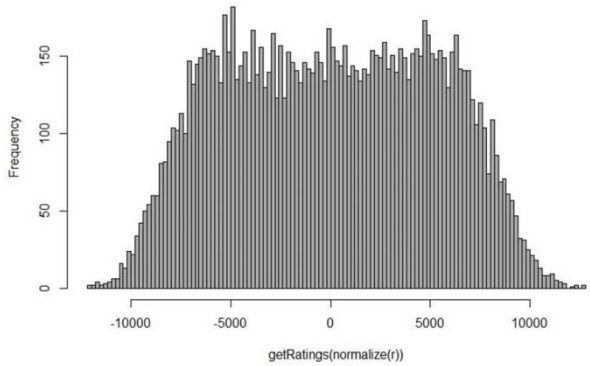

Figure 5: Histogram of getRatings(normalized(r))

The songs rated on average are skewed to the right but when normalized, we have a two – tail distribution. Here is why normalization works. It adjusts for the bias of individual raters. For example, a user might rate or play more songs more often than another user who usually rates or play it less. The normalization will take care of that user bias. The Figure above shows that the distribution of ratings is closer to a normal distribution after row centering and Z-score normalization additionally reduces the variance further.

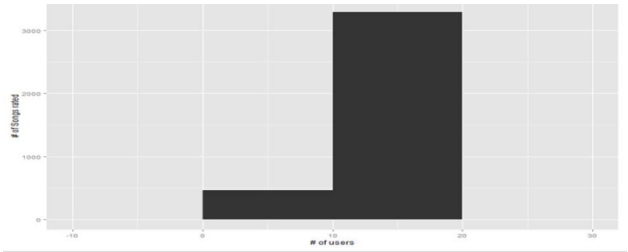

Figure 6: Songs rated on average

It seems that people don't play some music always and the long spike of the histogram shows that certain music was played more often than others.

To evaluate recommender systems, we will split the data into training and test sets. We will use the training set to build our model. We will hold out some items from the test set, and then make predictions using the model. Then we will compare our predictions against the holdout. If we predicted correctly, it is a true positive. If we predicted incorrectly, it is a false positive.

3.1 Evaluating and testing our model

We implemented four algorithms in this regard, that is Random, Popular Item, User-Based Collaborative Filtering, and Item-Based Collaborative Filtering.
The run time of each algorithm implementation is also displayed. For instance UBCF took longer time compared to other algorithms implemented; while IBCF took the lesser time to execute.

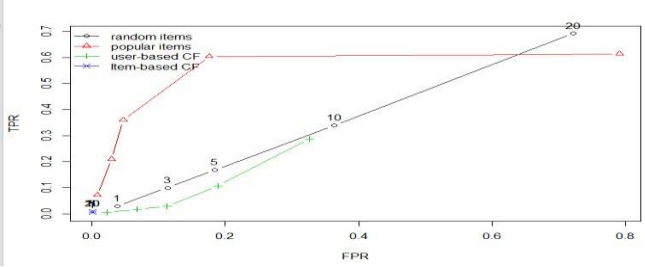

Figure 7: Comparison of ROC Curve for four algorithms used for the model

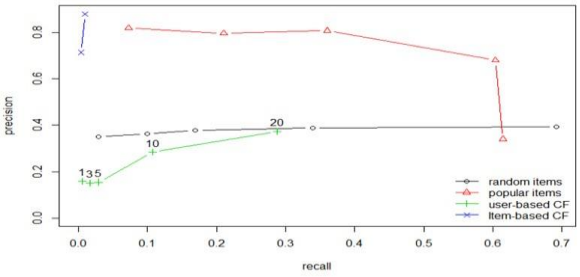

Figure 8: Comparison of the Precision and Recall of the four algorithms

Here, IBCF performed better than UBCF and all other algorithms. The algorithm which performed better lies in when and how are you generating recommendations. UBCF saves the whole matrix and then generates the recommendation at predict by finding the closest user. For large number of users-items, this becomes an issue. IBCF saves only k closest items in the matrix and doesn't have to save everything. It is pre-calculated and predicts simply reads off the closest items.
Predictably, RANDOM did better than UBCF perhaps surprisingly it seems, it's hard for RANDOM and UBCF to beat POPULAR. All this means that the songs rated high are usually liked by everyone and are safe recommendations. This might be different for other datasets.
In the ROC plot, we can see that UBCF did worse compare to other algorithm. The two best evaluation models are RANDOM and POPULAR; this may also be different from other dataset.

3.2 Clustering

The k-Means cluster was initially performed with 5 clusters and the songs predicted are shown in the table below:

Table 2: Predicted songs in R with k = 5

Cluster	Songs	Artist name
0	Funk you up	The Sequence
1	My own fault	Maria Taylor
2	Trial	Damien Jurado
3	Not the only one	Bonny Raitt

Table 3: K-means clusters and predicted songs with k =5

Cluster	Songs	Artist Name
0	Broken Silence	Better Luck next time
1	Do You Hear What I Hear	Suzy Bogguss
2	Your Little Hand	Gary Morris
3	Ellis Island Blues	Boo Hewerdine
4	Predominio del Sol	Nueva Vulcano

Both K-Means in R and Weka was compared to identify if their some commonalities and the figure below shows the comparison.

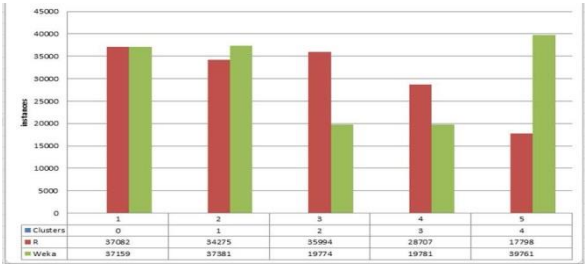

Figure 9: KMeans Cluster with k = 5

The same centroids are showing up in Weka for every value of k, whereas the centroids resulting from kMeans in language R are different every time. The cluster sizes resulting from the R analysis are more evenly distributed, whereas the cluster sizes in Weka have a wide variance as shown in chart above.

To make analyze more information in the clusters, we applied Expectation Maximization algorithm which uses probabilistic approach. Expectation–Maximization (EM) algorithm is an iterative method for finding maximum likelihood or maximum a posteriori (MAP) estimates of parameters in statistical models, where the model depends on unobserved latent variables.

Table 4: Expectation Maximization Clusters and Songs Predicted with K = 6

Clusters	Songs	Artist Name
0	One Way Mule	Silver chair
1	Waltz No. 6	Larry Coryell
2	Better Things	Bouncing Souls
3	Du Fehlst Mir So	Illegal 2001
4	The Storyteller Sleeps	Michael Gettel

Table 5: K-Means Clusters from Weka with K = 6

Clusters	Songs	Artist name
0	Broken Silence	Better Luck next time
1	Kennedy rag	Suzy Thompson
2	1 Piano Concerto No. 1 in B Flat Minor Op. 23: I. Allegro non troppo e molto maestoso	Emil Gilels
3	Rechtsstaat	Heideroosjes
4	Predominio del Sol	Nueva Vulcano
5	Escenas de la vida en el borde	Chango Spasiuk

Table 6: K-Means Clusters in R with K = 6

Clusters	Songs	Artist name
0	My Bike (Banana man album)	Ghoti Hook
1	Plug it in Otis	Bigod 20
2	Rowing	John Barry and John Debney
3	Za Horuca	Richard Muller
4	The house that faded	Blue Orchids
5	Ron Campbell	The Lamps

We compared the results obtained to identify which algorithm performed very well. The result can be seen in the figure below.

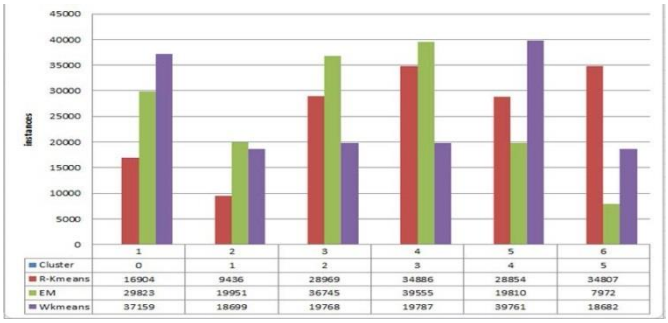

Figure 10: Kmeans for MSD with k=6

The result shows that Expectation Maximization and Clustering analysis in R have even distribution in their variance compared to Weka. The centroids are also of k-means in R and Expectation Maximization are always different at each iteration. However, the running time in Expectation Maximization is very much due to the large sets of data we used.

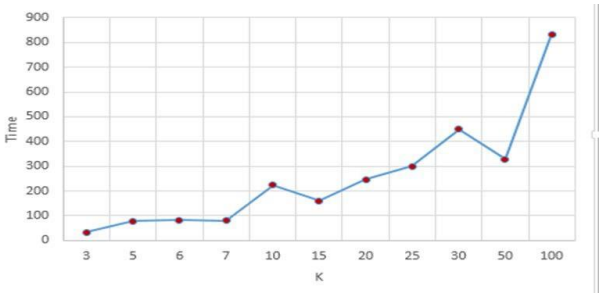

Figure 11: Comparison between k and time

From the plot we can see that the time of processing clusters increases when we keep increasing the value of k. However, this could be different based on the type of operating system the user deploys to build his recommendation. For this project, the researcher used a HP operating system with a 6 GB installed memory, Intel ® core(TM) i5, and 64 bit Operating system.

Other Approach

We also explored other approach in improving our system in order to minimize errors. So we compared Correlation, Moving Average Error and Root Mean Square Errors and from the plot result, we observed that RMSE is a better approach in predicting. The least method is Correlation as we can see from the plot.

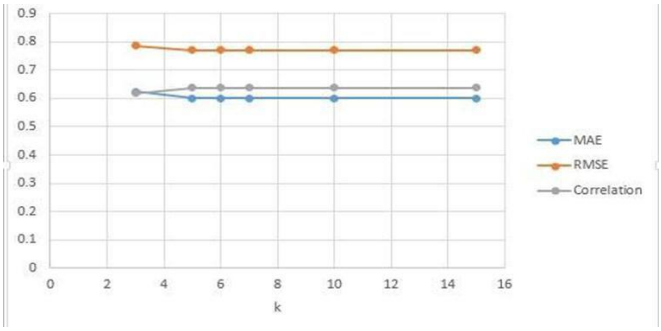

Figure 12: Comparing the Time of Running of each cluster

The Graph shows that RMSE performs better than MAE followed by correlation. Predicting Music using RMSE minimizes error than better than any other approach.

3.3 Social network analysis of million song data

The SNA of the Million Songs Dataset was also conducted in order to explore to which extent it is possible to use network analysis over implicit user feedback in order to learn or improve artist-artist associations. The data could be used for identifying metadata issues and/or enhancing the quality of the recommendations, via improved recall set and ranking function.

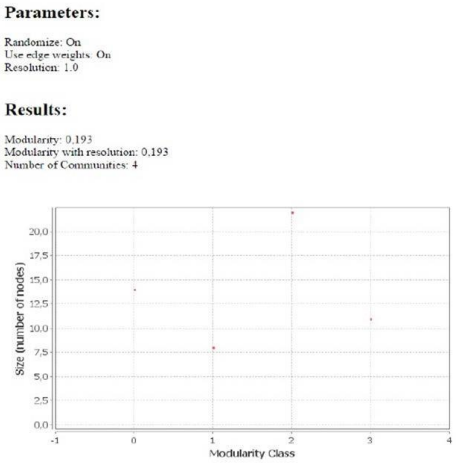

Figure 13: Size distribution

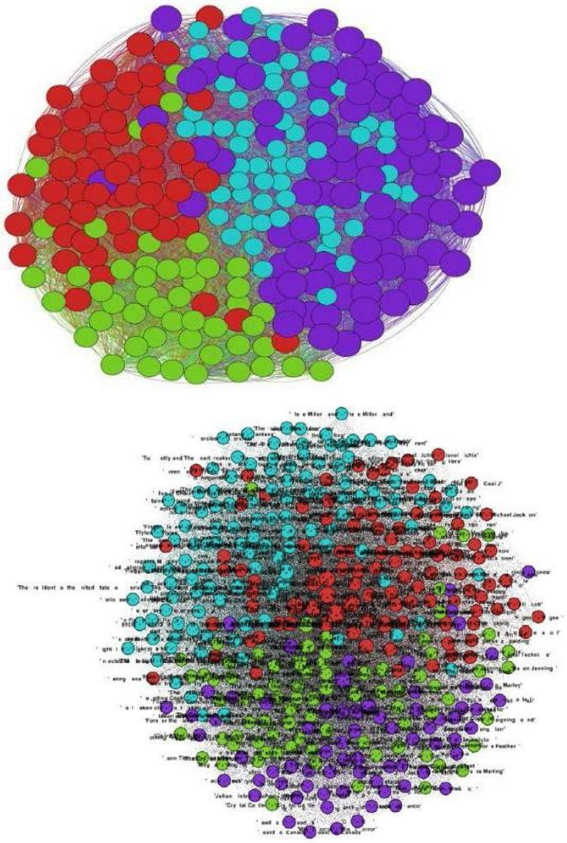

Figure 14: Visualization

From the above graph, we can see that there are communities or networks that exist among the artist. The networks show that these groups of artist sings either sing the same pattern of songs or have a common genre. The more we increase the Modularity, the more we see that artists form communities around different genres. Further works can be also conducted on the network analysis in order to study the clusters of songs and the rankings of each artist. It would be interesting to see the graph with the complete dataset from the Million Song Dataset. Community detection was particularly insightful: it surfaces artist compatibility, naturally from the interactions of users and music services. Besides that, degree and PageRank can be used for ranking popular artists inside communities, another useful piece of information for designing or improving music recommendation systems.

4. Results of findings

In this work, I have developed a recommendation system with Collaborative Filtering, and Content based approach using clustering such as k-Means and k-nearest neighbors' algorithm. The experiments using Million Song Dataset showed that recommending by k-NN is faster than recommending by Collaborative Filtering on any number of cores. Although the time frame increases as the clusters keep increasing. When more than one core is used, k-NN is faster. K-NN has also better scalability, but it is very sub-linear though.

Recommendations of Collaborative Filtering and k-NN are different – k-NN has better hit rate but Collaborative Filtering has better RMSE on hits. Clustering can decrease the needed time for a recommendation. It works even for small number of clusters – with five clusters the average time of recommendation decreased to less than half of the time that was needed by version without clustering. Unfortunately with increasing number of clusters the time didn't decrease much, because the k-Means algorithm created a lot of small clusters, but most of users kept in few largest clusters. We explored also the Social Network of Artists. In the social network, we tend to explore relationships that exist among the artists, and we found that indeed there is always a community of exist. This can be based on the genre of music. . The more we increase the Modularity, the more we see that artists form communities around different genres.

4.1 Conclusion

Recommender systems are a powerful new technology for extracting additional value for a business from its user databases. These systems help users find items they want to buy from a business. Recommender systems benefit users by enabling them to find items they like. Conversely, they help the business by generating more

sales. Recommender systems are rapidly becoming a crucial tool in E-commerce on the Web. Recommender systems are being stressed by the huge volume of user data in existing corporate databases, and will be stressed even more by the increasing volume of user data available on the Web 2.

New technologies are needed that can dramatically improve the scalability of recommender systems. As one of the most successful approaches to building recommender systems, collaborative filtering (CF) uses the known preferences [14] of a group of users to make recommendations or predictions of the unknown preferences for other users.

In this paper we explored CF-based recommender systems and compared it with other algorithms such as MAE, RMSE, and Correlation. Our results show that item-based techniques hold the promise of allowing CF-based algorithms to scale to large data sets and at the same time produce high- quality recommendations. This is similar to RMSE which performed better than MAE and correlation. Notwithstanding, CF uses RMSE method in making recommendation too.

However, for future works on large datasets, we recommend that SparkR be used since it is a distributed system. Spark framework is very easy to use more than Hadoop and the code is not complex, easy to read. The high speed and scalability of algorithms built on this system is good because it is embedded on Spark memory. SparkR can work faster for large datasets projects that require parallel solution but the scalability can be sub-linear, and very important characteristics are readability, easy deployment on many platforms because it is a distributed system and maintainability. It is also good for creating proofs of concepts, because creating a parallel program with Spark is nearly as easy as writing a sequential one.

References

1. http://labrosa.ee.columbia.edu/millionsong/
2. Goldberg, D., Nichols, D., Oki, B. M., and Terry, D. (1992). Using Collaborative Filtering to Weave an Information Tapestry. Communications of the ACM. December.
3. Konstan, J., Miller, B., Maltz, D., Herlocker, J., Gordon, L., and Riedl, J. (1997). GroupLens: Applying Collaborative Filtering to Usenet News. Communications of the ACM, 40(3), pp. 77-87.
4. Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., and Riedl, J. (1994). GroupLens: An Open Architecture for Collaborative Filtering of Netnews. In Proceedings of CSCW '94, Chapel Hill, NC.
5. Shardanand, U., and Maes, P. (1995). Social Information Filtering: Algorithms for Automating 'Word of Mouth'. In Proceedings of CHI '95. Denver, CO.
6. Hill, W., Stead, L., Rosenstein, M., and Furnas, G. (1995). Recommending and Evaluating Choices in a Virtual Community of Use. In Proceedings of CHI '95.
7. Breese, J. S., Heckerman, D., and Kadie, C. (1998). Empirical Analysis of Predictive Algorithms for Collaborative Filtering. In Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence, pp. 43-52.
8. Breese JS, Heckerman D, Kadie C (1998). "Empirical Analysis of Predictive Algorithms for Collaborative Filtering." In Uncertainty in Artificial Intelligence. Proceedings of the Fourteenth Conference, pp. 43–52.

9. Aggarwal, C. C., Wolf, J. L., Wu K., and Yu, P. S. (1999). Horting Hatches an Egg: A New Graph-theoretic Approach to Collaborative Filtering. In Proceedings of the ACM KDD'99 Conference. San Diego, CA. pp. 201-212.
10. Schafer, J. B., Konstan, J., and Riedl, J. (1999). Recommender Systems in E-Commerce. In Proceedings of ACM E-Commerce 1999 conference.
11. Demiriz A (2004). "Enhancing Product Recommender Systems on Sparse Binary Data."Data Mining and Knowledge Discovery, 9(2), 147–170. ISSN 1384-5810.
12. Deshpande M, Karypis G (2004). "Item-based top-N recommendation algorithms." ACM Transactions on Information Systems, 22(1), 143–177. ISSN 1046-8188.
13. Y. Hu, Y. Koren, and c. Volinsky, Collaborative filtering for implicit feedback datasets. In ICDM, pages 263-272, 2008.
14. Su, X, and Khoshgoftaar, T.M (2009). A survey of collaborative filtering techniques. In Adv. In Artif Intell. 2009.
15. Age of Big Data. Steve Lohr. New York Times, February 11, 2012 http://www.nytimes.com/2012/02/12/sunday-review/big-datas-impact-in-the-world.html
16. Hannon, J., Mike Bennett, M., Smyth, B. (2010). Recommending twitter users to follow using content and collaborative filtering approaches. RecSys 2010: 199-206.
17. Udeh Tochukwu Livinus, Rachid Chelouah, Houcine Senoussi, "Recommender System in Big Data Environment, Master's Thesis, 2015"

Udeh Tochukwu Livinus obtained his Master in Big Data and Data analysis at EISI in September 2015. Actually he is Business Analyst, and project research engineer, at Ever Christy Groups in Paris area, France. He has an international experience and exposure to diverse technologies, cultures, and business operations with more than 8 years' experience.

Rachid Chelouah He received first his engineering degree in network and telecommunication in 1988, the PhD degree from the University of Cergy, in 1999 and the Doctorate of Sciences (Habilitation) from the University of Cergy, in 2014. He was first, Project Manager at Net2S during 2 years; he led development of mobile telephony platform, and after he joined Dassault as research engineer. In 2000 he left Dassault to EIVD, high engineering school in Switzerland for 5 years. Since 2005 he joined the EISTI to become head of the IT department from 2006 to 2016 and he was named research director in 2014. His main research interests are data scientist methods and their applications in various fields of IT engineering.

Houcine Senoussi is an associate Professor at EISTI. His main research interests are in fields of recommender systems and semantic web.