Music Recommendation System Based on Ratings Obtained from Amazon

Sergio Marcos-Vazquez, Carlos Fernandez-Lozano, Adrian Carballal, and Francisco Cedron

RNASA-IMEDIR, Faculty of Computer Science, Universidade da Coruña, 15071 A Coruña, Spain
Software Engineering Laboratory, Faculty of Computer Science, Universidade da Coruña, 15071 A Coruña, Spain
Centro de Investigación CITIC, Universidade da Coruña, 15071 A Coruña, Spain
Correspondence: francisco.cedron@udc.es

DOI: *https://doi.org/10.17979/spudc.000024.03*

Abstract: In the current context of an era in which a significant portion of people are constantly living online, with various multimedia streaming platforms serving as major sources of entertainment, and with e-commerce playing also a key role, recommender systems are carving out their place as one of the most important and widely used tools for enhancing user experiences on these platforms. This work undertakes a comparative study on some of the techniques used within these systems, mainly focused on those based in collaborative filtering. Multiple recommender systems will be implemented according to each of these methods, taking for this purpose the vinyl records and CDs Amazon's user ratings.

1 Introduction

According to the Recording Industry Association of America, the music industry business has been on an upward trend since 2014 (Recording Industry Association of America, 2023). This may be due to the data provided by the billboard indicating that more than 100,000 songs are uploaded every day(Billboard, 2023).

With such a large number of songs, it is understood that it is impossible for a person to manually select which songs they like the most manually. It is extremely necessary to have a recommendation system for some time that indicates users new songs based on the tastes that it leaves in their musical fingerprint.

One way to obtain a person's personalized tastes is through the reviews they publish in ecommerce. With all the data found in ecommerce, profiles of aesthetic tastes could be established that help propose new content to users based on the evaluations of users who have a similar musical taste.

In this work, reviews of CDs and vinyl obtained from Amazon will be used (Rappaz et al., 2021). The objective is to use the ratings in several recommendation systems based on collaborative filtering, measuring and comparing the results obtained for each of them.

2 Collaborative filtering systems

Collaborative filtering recommendation systems are based on the evaluation of the interest a user may have in a certain item based on the opinions of other users(Schafer et al., 2007). This is based on the premise that if two users have similar opinions on certain items, it is likely that an item liked by one of the users will also be liked by the other.

In order to capture the preferences and behaviours of users with respect to items, a matrix is created that has as dimensions users and items, and where the value of the cells would correspond to the ratings. This matrix is fundamental in recommendation systems based on collaborative filtering, as it will be used as the basis for the algorithms used in this type of recommendation systems.

These matrices tend to have a peculiarity: it is normal that a user does not rate all the articles, nor is an article rated by all the users. In fact, the number of ratings that a user usually makes is tiny in comparison to the number of articles that he or she can rate, and the same is true for articles, where the number of ratings received is also very small in comparison to the number of existing users. This causes most of the cells of the rating matrix to be empty, generating sparse matrices(Branham, 1990).

Depending on how you work with this matrix, you can find two categories into which collaborative filtering recommendation systems can be divided: collaborative filtering based on models and collaborative filtering based on neighbors(Singh et al., 2020).

2.1 Collaborative filtering based on models

This category attempts to fill in the item-user rating matrix, predicting the ratings that a user would give to an item that they have not rated at the moment. These models are built from a training dataset that contains patterns with item-user relationships.

Alternating Least Squares

The Alternating Least Squares (ALS) method is an algorithm used for matrix factorization. It factors a given matrix R into two smaller matrices U and V such that $R \approx U^T V$(Hastie et al., 2015). The goal is to predict user preferences for items based on past interactions. The algorithm attempts to estimate the ratings matrix R as the product of two lower-rank matrices, X and Y, i.e., $X * Yt = R$(Clarkson and Woodruff, 2017). During each iteration, one of the factor matrices is held constant, while the other is solved for using least squares. The newly-solved factor matrix is then held constant while solving for the other factor matrix.

Singular Value Decomposition

The singular value decomposition (SVD) of an array is a factorization of that array into three arrays $A = UWV^T$(Banerjee and Roy, 2014). U is mxn array of the orthonormal eigenvectors of AA^T. V^T is the transpose of a nxn array containing the orthonormal eigenvectors of $A^T A$. Finally W is an nxn diagonal array of the singular values that are the square roots of the eigenvalues of $A^T A$.

Non-negative Matrix Factorization

Non-negative matrix factorization (NMF) is a group of algorithms in multivariate analysis and linear algebra where a matrix V is factorized into two matrices W and H, with the property that all three matrices have no negative elements(Sra and Dhillon, 2005). This non-negativity makes the resulting matrices easier to inspect.

In NMF, the goal is to find two non-negative matrices W and H whose product approximates the non-negative matrix X. This factorization can be used for example for dimensionality reduction, source separation or topic extraction.

2.2 Collaborative filtering based on neighbors

This category is based on using various similarity metrics on two vectors of the item-user matrix, thus calculating a similarity with which to find the nearest neighbors.

k-Nearest Neighbor

The k-nearest neighbors algorithm (k-NN) is a non-parametric supervised learning method(Cover and Hart, 1967). In k-NN, the input consists of the k closest training examples in a data set. The output depends on whether k-NN is used for classification or regression. In k-NN classification, the output is a class membership. An object is classified by a plurality vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors. If $k = 1$, then the object is simply assigned to the class of that single nearest neighbor. In k-NN regression, the output is the property value for the object. This value is the average of the values of k nearest neighbors. If $k = 1$, then the output is simply assigned to the value of that single nearest neighbor.

3 Methodology

The dataset has 4,543,369 ratings, made by a total of 1,944,316 users on 434,060 products. Not all of this data will be used to train the recommendation system, but only the ratings made by users with 10 or more ratings will be used, which leaves the dataset to be used with 1,351,025 ratings and 44,875 users. This filtering is carried out in order to try to avoid the cold start problem(Tey et al., 2020).

With the data prepared, and before starting to train the model, 20% of the ratings are extracted from the dataset, which will be used as a validation set to be able to check the good fit and performance of the model during training. This division occurs in a stratified manner(Ojala and Garriga, 2010). For training, 10-fold cross-validation is used in order to prevent overfitting. The average value of the 10 folds is used as the result.

To adjust the algorithms, we proceed in the same way for all of them, using the hyperparameter adjustment technique known as grid search(Yang and Shami, 2020). The values used are shown in table 1.

Table 1: Values used in the grid search technique by the different collaborative filtering methods. The meaning of each hyperparameter can be consulted in the documentation of the implementation used. ALS: Apache Spark - MLlib (2022), SVD: Nicolas Hug (2020c), NMF: Nicolas Hug (2020b) and k-NN: Nicolas Hug (2020a).

Method	Parameter	Values
ALS	MaxIter	5, 10, 15, 20
	Rank	10, 15, 25, 30, 50, 100
	RegParam	0.1, 0.3, 0.5, 0.7, 1.5
SVD	N_factors	50, 100, 200
	N_epochs	50, 100, 200
	Lr_all	0.005, 0.01
	Reg_all	0.02, 0.05, 0.1
NMF	N_factors	15, 50, 100, 200
	N_epochs	25, 50, 75, 100, 200
	Reg_pu	0.06, 0.1, 0.3
	Reg_qi	0.06, 0.1, 0.3
k-NN	K	10, 20, 25, 30, 40, 70, 100, 1000, 1163, 1500, 10000

Finally, the metrics used to measure the performance of the different methods were the following:

- **MAE**: Mean absolute error calculates the average of the absolute differences between the predicted ratings and the actual ratings.
- **RMSE**: The root mean square error measures, like the MAE, the difference between the predicted and actual valuations, with the difference that, instead of using absolute differences, it calculates the squared differences and takes the square root of the average of This differences. By squaring the differences, this metric penalizes larger errors in predictions more.
- **Precision**: Precision is defined as the ratio of relevant recommendations within the top k of recommendations over the total of recommended articles within the top k, taking in this specific case as relevant recommendations those that have a rating greater than 4.0, and 10 as the value of k.
- **Recall**: Recall is defined as the ratio of relevant recommendations within the top k of recommendations over the total of relevant articles, taking as relevant those ratings greater than 4.0 and 10 as the value of k, as in precision.
- **F1-Score**: Combination of precision and recall measures into a single value, allowing the combined performance of both to be compared.

4 Results

In order to know the performance achieved by each of the different methods, the best model obtained by each method will be used. Table 2 shows which hyperparameters of each algorithm have obtained the best results.

Table 2: Hyperparameters of each method that obtained the best results.

Method	Hyperparameters
ALS	maxIter=20, rank=10, regParam=0.3
SVD	n_factors=200, n_epochs=100, lr_all=0.005, reg_all=0.1
NMF	n_factors=50, n_epochs=50, reg_pu=0.1, reg_qi=0.06
k-NN	k=100

The results obtained by the different collaborative filtering methods are shown in table 3.

Table 3: Hyperparameters and execution time of each method that obtained the best results.

Method	RMSE	MAE	Precision	Recall	F1-Score	Execution time
ALS	0.97761	0.76676	0.23682	0.98334	0.38172	1h 45 min
SVD	0.83394	0.57884	0.85034	0.83183	0.84098	9h 30 min
NMF	0.92317	0.58730	0.85382	0.82332	0.83829	2h 50 min
k-NN	0.96502	0.74615	0.86629	0.94585	0.90432	30 min

5 Conclusions

The ALS algorithm is the one that obtains the worst results, having similar RMSE and MAE to KNN. This is surprising since ALS is an algorithm focused on predictions, while KNN focuses on recommendations, which can be seen at first glance since it is the algorithm that obtains the best F1-Score. Of course, the F1-Score of ALS, which is very low compared to the rest, is entirely due to the precision that this algorithm has, since its recall is the best of all, the reason for this being that the algorithm focuses in recommending the most popular products. On the other hand and as a point in favor, although the RMSE and MAE values obtained are the worst compared to the rest, they are not considered bad results, and combining this with the fact that the combination of execution time of this algorithm competes with Because NMF is the best, ALS can be considered a good option if you want to obtain acceptable recommendations in a very good time.

Regarding KNN, as already mentioned, it gives the best F1-Score, having slightly the highest precision plus a very good recall. This makes a lot of sense since it is an algorithm especially focused on recommendations, but the RMSE and MAE it obtains are also quite acceptable.

For the two remaining algorithms, and starting with SVD, it can be seen that of all the algorithms it is the one that obtains the best results overall.

On the other hand, it is also very clear that this algorithm is the slowest, and its use could be seen in cases where the best possible recommendations are wanted without the speed of obtaining them being a very important factor. Finally, the NMF algorithm is consolidated as a very solid option, since the difference in results with SVD is very small, even achieving better precision. It can be criticized that it can make some larger errors than SVD as can be seen in its RMSE values, but this is still a very good value. Also, its execution time is quite decent, more than good when compared to SVD.

Acknowledgements

CITIC is funded by the Xunta de Galicia through the collaboration agreement between the Consellería de Cultura, Educación, Formación Profesional e Universidades and the Galician universities for the reinforcement of the research centres of the Galician University System (CIGUS).

We are particularly grateful to CESGA (Galician Supercomputing Centre) for providing access to their infrastructure.

Bibliography

Apache Spark - MLlib. Als - pyspark 3.4.1 documentation. *https://spark.apache.org/docs/latest/api/python/reference/api/pyspark.ml.recommendation.ALS.html*, 2022. [Online; accessed 13-September-2023].

S. Banerjee and A. Roy. *Linear algebra and matrix analysis for statistics*. Chapman & Hall/CRC Texts in Statistical Science. Chapman & Hall/CRC, Philadelphia, PA, June 2014.

Billboard. The ledger: Are there really 100,000 new songs uploaded a day? maybe more. *https://www.riaa.com/u-s-sales-database/*, 2023. [Online; accessed 13-September-2023].

R. L. Branham. Sparse matrices. In *Scientific Data Analysis*, pages 34–66. Springer New York, 1990.

K. L. Clarkson and D. P. Woodruff. Low-rank approximation and regression in input sparsity time. *Journal of the ACM*, 63(6):1–45, Jan. 2017.

T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27, 1967.

T. Hastie, R. Mazumder, J. D. Lee, and R. Zadeh. Matrix completion and low-rank SVD via fast alternating least squares. *Journal of Machine Learning Research*, 16:3367–3402, 2015.

Nicolas Hug. Surprise: A python library for recommender systems - k-nn inspired algorithms. *https://surprise.readthedocs.io/en/stable/knn_inspired.html*, 2020a. [Online; accessed 13-September-2023].

Nicolas Hug. Surprise: A python library for recommender systems - matrix factorization-based algorithms - nmf. *https://surprise.readthedocs.io/en/stable/matrix_factorization.html#surprise.prediction_algorithms.matrix_factorization.NMF*, 2020b. [Online; accessed 13-September-2023].

Nicolas Hug. Surprise: A python library for recommender systems - matrix factorization-based algorithms - svd. *https://surprise.readthedocs.io/en/stable/matrix_factorization.html#surprise.prediction_algorithms.matrix_factorization.SVD*, 2020c. [Online; accessed 13-September-2023].

M. Ojala and G. C. Garriga. Permutation tests for studying classifier performance. *Journal of machine learning research*, 11(6), 2010.

J. Rappaz, J. McAuley, and K. Aberer. Recommendation on live-streaming platforms: Dynamic availability and repeat consumption. In *Fifteenth ACM Conference on Recommender Systems*. ACM, Sept. 2021.

Recording Industry Association of America. U.s music revenue database. *https://www.riaa.com/u-s-sales-database/*, 2023. [Online; accessed 13-September-2023].

J. B. Schafer, D. Frankowski, J. Herlocker, and S. Sen. Collaborative filtering recommender systems. In *The Adaptive Web*, pages 291–324. Springer Berlin Heidelberg, 2007.

P. K. Singh, P. K. D. Pramanik, and P. Choudhury. Collaborative filtering in recommender systems: Technicalities, challenges, applications, and research trends. In *New Age Analytics*, pages 183–215. Apple Academic Press, May 2020.

S. Sra and I. Dhillon. Generalized nonnegative matrix approximations with bregman divergences. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems*, volume 18. MIT Press, 2005.

F. J. Tey, T.-Y. Wu, C.-L. Lin, and J.-L. Chen. Accuracy improvements for cold-start recommendation problem using indirect relations in social networks. *Journal of Big Data*, Dec. 2020.

L. Yang and A. Shami. On hyperparameter optimization of machine learning algorithms: Theory and practice. *Neurocomputing*, 415:295–316, Nov. 2020.