Matrix Factorization Technique for MovieLens Recommender System

Shilpa Balan
College of Business and Economics, Department of Information Systems
California State University, Los Angeles
sbalan@calstatela.edu

Pamella Howell
College of Business and Economics, Department of Information Systems
California State University, Los Angeles
phowell@calstatela.edu

Yash Choksi
College of Business and Economics, Department of Information Systems
California State University, Los Angeles
ychoksi@calstatela.edu

Abstract. Users have online access to millions of audio tracks and movies. Online streaming platforms widely use AI-based recommender systems to help users choose the songs to listen to and the movies to watch. As the volume of available content rises, a standardized methodology to evaluate recommender systems is required. This paper focuses on collaborative-based filtering method of recommender systems. Leveraging the matrix factorization technique, we provide comprehensive information on an algorithm for improving prediction accuracy using standardized Python code and validated with the MovieLens data set.

Introduction

Researchers estimate that retail e-commerce sales will increase from 3.5 billion in 2019 to 6.5 billion in 2023 (Clement, 2020). The growth of e-commerce has flooded customers with product choices. Consumers inundated with product offerings have to assess what options will meet their needs. As a result, customers face an increased cognitive burden to process data and gather meaningful information for making purchase decisions (Kim, Kim, & Lee, 2000; B. Schafer, Konstan, & Riedl, 2001). Businesses that pair consumers with content meeting their needs will increase satisfaction and retention. Therefore, recommender systems are useful to retailers and content providers.

Recommender systems are an algorithmically based technique for providing suggestions to users. They personalize suggestions based on a user's taste by analyzing their product interest. Recommendations made to users help in various ways, such as deciding what items to buy, what music to listen to, or what news to read (Ricci, Rokach, & Shapira, 2011). The increased inclination of customers to purchase or utilize services will significantly impact a business's revenue. Recommender systems may also influence users' trust in the systems and the click-through rate, though the latter is debated among researchers (Zheng et al., 2010).

Due to the increased advantages of personalized recommendations, e-commerce leaders like Amazon and Netflix have integrated recommender systems as an integral part of their websites (Koren et al., 2009). Recommender systems also play an essential role in highly rated internet sites such as YouTube, Yahoo, Tripadvisor, and IMDb, to name a few. Many media companies deploy recommender systems as part of the services they provide to their subscribers (Ricci et al., 2011).

With the information overload on the web, recommender systems have proven to be valuable to online users. Correspondingly, organizations and data scientists have explored various techniques for recommendation generation. The traditional categorizations for recommender systems include collaborative-based filtering, content-based filtering and a hybrid method that combines content and collaborative filtering (Adomavicius & Tuzhilin, 2005). Collaborative filtering is the most successful recommendation technique; it is used for recommending web pages, movies, articles, and products (Shardanand & Maes, 1995).

Collaborative filtering identifies customers whose interests are similar and recommends products based on their peers' historical preferences (Cho, Kim, & Kim, 2002). The quality of these recommendations play a vital role in the consumer's online shopping behavior. A keen area of interest among researchers has been movie recommendations. This topic continues to drive interest due to the disruptive nature of innovations (Min & Zhu, 2014).
For example, future changes in online movie sales may lead to the reorganization of the movie theater industry.

The Netflix Prize Challenge, conducted in 2006, was an open competition designed to reward the best collaborative filtering algorithm for predicting user ratings of movies. Simons Funk, the winner of the Netflix Prize Challenge competition utilized Singular Value Decomposition (SVD) in his recommender systems algorithm. Funk made his algorithm available via a Python library; some of the code is available via GitHub (Funk, 2020). Another scientist, Nicolas Hug, compiled his recommender systems algorithm and made it available via the Python scikit package called 'Surprise' (Hug, 2015).

The Netflix Prize Challenge inspired us to document the Python application of SVD in Recommender Systems. We extend their work by providing documentation of the SVD application for recommender systems in an easy-to-understand format with the code and mathematical mapping. In this paper, we do not use the SVD libraries created by Funk or Hug and we demonstrate the application of SVD using the MovieLens data set.

Furthermore, a literature survey, examining two hundred articles over 16 years, found that most authors failed to provide detailed information about their algorithms (Beel et al., 2016). It is, therefore, increasingly difficult to validate variations or improvements in the accuracy of recommender algorithms. This paper fills the literature gap by providing researchers and businesses with a fully validated algorithm based on the matrix factorization technique using Python packages. To the best of our knowledge, this paper is one of the first academic publications to provide detailed information about the SVD algorithm for recommender systems by applying the matrix factorization technique.

The rest of the paper is organized as follows. First, we review the existing literature. Second, the methodology section describes our approach to building a recommender systems algorithm. Third, we describe and discuss the algorithm and finally close with a conclusion and a discussion of future research.

Background

A recommendation approach is a model that helps determine how recommendations could be provided to a user. The recommendation scenario is based on domain and user characteristics (Beel et al., 2016). The two main types of recommender systems are content-based filtering and collaborative-based filtering. Content-based filtering systems learn to recommend items similar to those that the user liked in the past. The similarity of the items is computed based on the features associated with the compared items. For example, if a user has positively rated a movie from the comedy genre, the system can learn to recommend other movies from this genre (Ricci et al., 2011).

In contrast, collaborative filtering systems recommend items that other users with similar tastes liked in the past (J. B. Schafer, Frankowski, Herlocker, & Sen, 2007). For example, in a collaborative prediction movie recommendation system, the inputs to the system are the ratings of the movies that the users have already viewed. The patterns from these ratings given by other users help determine a movie (not seen yet) for a user to watch. This process can be formalized as a matrix completion problem (Azar et al., 2001; Hofmann, 2004). An approach to collaborative prediction is to fit a factor model to the data and use it for predictions (Canny, 2004; Marlin & Zemel, 2004).

The term collaborative filtering was coined in 1992. Users collaborate by sharing their ratings. By providing ratings, information filtering is more useful since humans are involved in the filtering process (Goldberg, Nichols, Oki, & Terry, 1992). With this methodology, like-minded users are identified, and items that one user rated positively are recommended to another user (Beel et al., 2016). Collaborative filtering is conducted using two popular methods, the latent approach upon which matrix factorization is based and the neighborhood method. Latent models infer ratings from user and item factors (Koren et al., 2009).

When using the neighborhood-based approach, the prediction is computed by using the item similarity weights. The top items having the highest predicted ratings are recommended to the user. Using the Netflix data set, previous researchers applied the user-based collaborative filtering approach and looked for the most similar users for the current user. Previous researchers concluded that using the neighborhood-based approach was better when the number of users is far greater than the number of items (Ponnam et al., 2016).

The matrix factorization approach is favored because it produces more successful results than the neighborhood method (Koren et al., 2009). A vital feature of the matrix factorization approach is the ability to infer missing attributes or previous user or item patterns. The matrix factorization technique also handles data with high dimensionality. Information such as the users' preferences, geographical information, calendar information, or social networking data can be exploited to create intelligent recommendations (Ballatore et al., 2010). With data available from multiple sources using matrix factorization can result in highly customized recommendations.

A popular algorithm of the matrix factorization technique was provided by Simons Funk, the winner of the Netflix Prize Challenge, that was conducted in 2006. Based on information available via Github, we infer that the Numba compiler can be used to increase the speed of Funk's algorithm (Funk, 2020). Improved processing speed is another benefit of using the matrix factorization technique. Funk's algorithm is claimed to be comparably faster than similar algorithms, such as Nicolas Hug's Surprise (library made available via Python) algorithm (Funk, 2020). The literature on speed performance is sparse. A full discussion of the performance measures based on speed is hence outside the scope of this study.

Previous studies have examined recommender systems using the MovieLens data set. One study used the MovieLens data to analyze the effect of item recommendations on users' opinions. To do so, the authors evaluated different rating scales, such as binary (thumbs up or thumbs down), no-zero (a scale from -3 to +3 with no zero), and half-star (a 0.5 to 5-star scale in half-star increments) (Cosley et al., 2003). Using 15 different movies on each scale, they examined how new ratings mapped to the original ratings.

In another study aimed at integrating the MovieLens and the IMDb data set, researchers faced several difficulties in data cleaning. The authors detected several anomalies in the MovieLens data set, such as duplicate titles, unknown, and null (Peralta, 2007). The SVD approach demonstrated in this paper is robust enough the combat data anomalies. Similarly, in an environment where companies have access to unstructured data sets with irregularities, the matrix factorization approach can be leveraged advantageously.

The methodology section details the pseudocode we created using the matrix factorization approach

Methodology

Figure 1 shows the methodology we applied to our study. The data set comprising multiple user profiles is first structured. Matrix Factorization using SVD (Singular Value Decomposition) is then applied to make movie recommendations for a user.

Data

MovieLens is a recommender system that asks its users to provide movie ratings for personalized movie recommendations (MovieLens, 2020). The data set was first released in 1998 and has been downloaded numerous times. This is partly due to the phenomenal rate of growth of recommendation research. MovieLens has averaged 20 to 30 new user sign-ups every day (Gladwell, 2000). In mid-2005, MovieLens added the ‘movie detail’ pages into their site design. The addition of movie tagging in late 2005 provided objective and subjective data for describing movies (Harper & Konstan, 2015).

This study uses the MovieLens data set generated on October 17, 2016. The data describes 5-star rating and free-text tagging activity from MovieLens (MovieLens, 2020). The data contains 100,004 ratings, 1,296 tag applications across 125 movies (MovieLens, 2020). The data was created by 671 users between January 09, 1995, and October 16, 2016 (Harper & Konstan, 2015). The data describes people’s preferences for movies on a 0 - 5 star scale for each movie (Gladwell, 2000; Yeung, 2010). Users who had rated at least 20 movies were selected at random. There is no demographic information in the data.

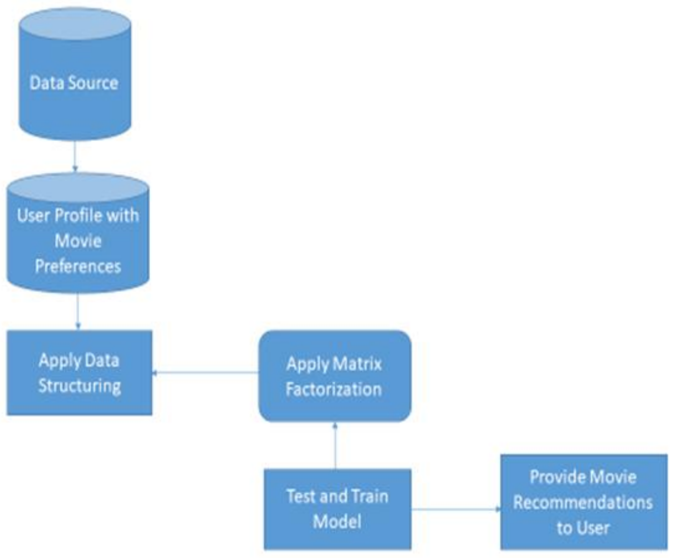

Data Source
User Profile with Movie Preferences
Apply Data Structuring
Apply Matrix Factorization
Test and Train Model
Provide Movie Recommendations to User

Figure 1: Methodology

Algorithm: Matrix Factorization

Recommendations can be generated by a wide range of algorithms (Koren et al., 2009). The Netflix Prize competition demonstrated that matrix factorization models are superior to nearest-neighbor techniques for producing product or item recommendations. While user-based or item-based collaborative filtering methods are simple, matrix factorization techniques are usually more effective because they allow us to discover the latent features that is central to the interactions between users and items.

Matrix factorization is a mathematical approach to highlight patterns hidden in the data (Yeung, 2010). In the matrix form, each row represents each user, while each column represents different movies. Using matrix factorization, we can estimate if a user would like a movie that the user has not seen yet. And if that estimation is high, we can recommend that movie to the user (Koren et al., 2009).

Singular Value Decomposition (SVD) is a technique for dimensionality reduction (Golub & Reinsch, 1971). It is also related to Principal Component Analysis (PCA). Principal Component Analysis reduces a dataset of dimension n to dimension k (k<n). The core of the SVD algorithm lies in the following theorem: It is always possible to decompose a given matrix A into A =UλVT, where VT is the transpose of V (Ricci et al., 2011). Given the n×m matrix data A (n users, m movies), we can obtain an n×r matrix U (‘n’ users, ‘r’ eigenvectors), an r×r diagonal matrix λ (the singular values), and an m×r matrix V (‘m’ movies, ‘r’ eigenvectors). Based on the SVD theorem, we can thus state:

$$\boxed{A_{nxm} = U_{nxn} \lambda_{nxm} V^{T}{}_{mxm}}$$

“Eigenvectors and eigenvalues are also referred to as characteristic vectors and latent roots or characteristic equation (in German, eigen means “specific of” or “characteristic of”)” (Abdi, 2007). Eigenvalues are strong predictors of the quality of recommendations. The magnitude of the eigenvalue is strongly correlated with the accuracy of recommendations for that user (Sarwar, Karypis, Konstan, & Riedl, 2000).

In order to compute the SVD of a matrix A, we consider AAT and ATA, where AT is the transpose matrix of A. The columns of U are the eigenvectors of AAT, and the columns of V are the eigenvectors of ATA. The singular values on the diagonal of λ are the positive square roots of the non-zero eigenvalues of both AAT and ATA (Sarwar et al., 2000; Zheng et al., 2010). The original matrix A can be approximated by truncating the eigenvalues at a given latent factor (Sarwar et al., 2000).

SVD can be used to uncover latent relations between customers and products. Figure 2 depicts the SVD technique for matrix factorization we applied in our study. We implemented this technique in Python using packages and libraries provided.

```
Import Python libraries pandas, numpy, matplotlib.pyplot and scipy.sparse.linalg;
Create a data frame for ratings titled ratings_dataframe;
Create a data frame for movies title movies_frame;
Convert ratings_dataframe to a matrix format using as_matrix() in Python;
Normalize the ratings matrix by each user's mean using the numpy package in Python to generate a new ratings matrix;
Use svds (singular value decomposition) function on the new ratings matrix;
Verify the 3 matrices generated:
        User matrix, Singular Value matrix and Transpose of Ratings matrix (U, S and $V^T$);
Set the number of latent factors;
Compute the prediction as dot product of U, S and $V^T$;
Recommend movies for each user based on prediction:
        Return the movie with the highest rating that the user has not rated yet;
```

Figure 2: Pseudocode of the Recommender Systems Algorithm

Discussion

We implemented our algorithm for Recommender System packages and libraries provided by Python. Figure 2 shows the pseudocode of our algorithm. The Python packages used are:

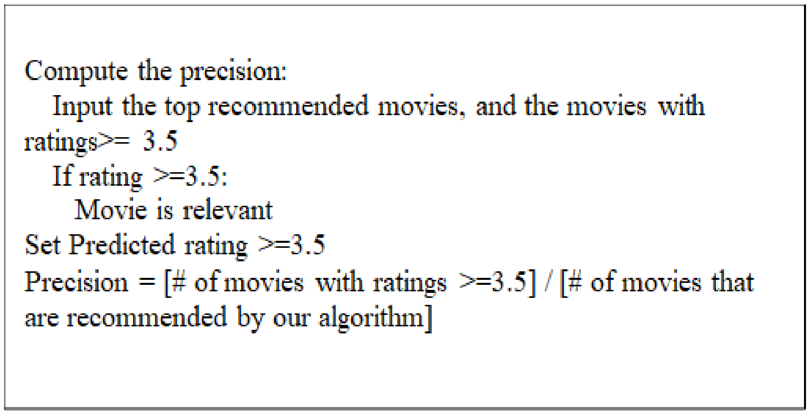
- `Numpy`: This is an open source numerical and matrix computation package for Python. Since our algorithm involved matrix computation, `numpy` is one of the methods to accomplish this.
- `Pandas`: This is a data processing package, and helps in handling data in Python.
- `Matplotlib`: This package is used to create visualizations in Python.
- `Scipy`: This package is developed on top of `numpy`. We computed Singular Value Decomposition using this package. SVD is part of linear algebra, and hence we used sub package, the '`linalg`'

The pseudocode of our algorithm mapped to its mathematical notations are described in Table 1. We referred to Edel Garcia's SVD mathematical tutorial to help map the Python code with the mathematical notations (Garcia, 2006).

As indicated in Table 1, *k* is the number of latent features in our pseudocode. For movies, predictions from lower rank matrices with values of *k* between roughly 20 and 100 have been found to be the best at generalizing to unseen data (Becker, 2020). Further, we computed the precision of our algorithm, as shown in Figure 3. Precision is computed using the following formula (Bondarenko, 2019):

$$Precision = (recommended \cap relevant) / recommended$$

The resulting precision is a perfect score of 1.0. In the recommender system, we set a threshold for the relevant ratings to compute the algorithm's precision. The threshold for movie ratings we set as greater than or equal to 3.5. When evaluating movie recommendations, the recall value is of not much importance, as it can vary based on the number of recommended movies at a given time.



```
Compute the precision:
   Input the top recommended movies, and the movies with
ratings>= 3.5
   If rating >=3.5:
      Movie is relevant
Set Predicted rating >=3.5
Precision = [# of movies with ratings >=3.5] / [# of movies that
are recommended by our algorithm]
```

Figure 3: Evaluation of the Recommender Systems Algorithm

Python Pseudocode	**Mathematical Notation/ Description**
Create ratings matrix	*R* is user ratings matrix
De-mean the data	Normalize the ratings matrix by each user's mean using the numpy package to generate a new ratings matrix; *Set R-new=New ratings matrix*
Set the number of latent factors	Set *k*= number of latent features. We can think of *k* as the most important underlying preference vectors.
from scipy.sparse.linalg import svds	*U, S,* V^T = svds (*R*-new, *k*=50) • *U is the user "features" matrix.* • *U represents how much users "like" each feature* • V^T *is the movie "features" matrix* • *k: Number of latent features.*
svds function in Python used to generate the singular values	• Compute the transpose of *R*-new (*R*-new=*A*) and A^TA • Determine the eigenvalues of A^TA and sort these in descending order (in the absolute sense). • Get eigenvalues *c1* and *c2*. • Square root the eigenvalues to obtain the singular values of *A*. Get singular values *s1* and *s2* by taking the square root of the eigenvalues.
Verify the matrices generated	• Construct diagonal matrix *S*: place singular values in descending order along its diagonal. • Compute its inverse, S^{-1} . • Use the ordered eigenvalues and compute the eigenvectors of A^TA. • Place these eigenvectors along the columns of *V* and compute its transpose, V^T . • Compute *U* as $U = AVS^{-1}$

Table 1. Mapping of the Pseudocode to the Mathematical Notation of SVD

Conclusion and Future Research

Recommender systems are applied to a wide range of problem domains, including books, electronic media, and entertainment. By using recommender systems, businesses can generate personalized recommendations based on user preferences. The advancements of recommender systems gained momentum with companies' willingness to share data. For example, the Netflix Prize Challenge led to the refinement of the matrix factorization technique. Matrix factorization techniques are a dominant methodology within collaborative filtering recommenders. Collaborative-filtering utilizes past users with similar tastes to make recommendations to a given user.

The focus of our analysis is not to invent a new recommender technique. We aim to extend current documentation and to combat the variability of research results. We develop and explicate a Python-based algorithm; by doing so, we extend the available documentation matrix factorization. We also found that matrix factorization produces high accuracy and significantly reduces dimensionality in large datasets. Researchers and businesses can apply the pseudocode from this study to their recommender system applications and validate improvements in model performance.

In future research, one can apply the matrix factorization technique to other datasets or contexts, such as, healthcare recommendations. Researchers can examine the effect of user dimensions like a person's social role or relationships on recommendations given. Additional research is also needed on algorithms for producing coherent sequences of recommendations.

References

Abdi, Hervé. "The Eigen-Decomposition: Eigenvalues and Eigenvectors." *Encyclopedia of measurement and statistics*, 2007, pp. 304-308.

Adomavicius, G. and A. Tuzhilin. "Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions." *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 6, 2005, pp. 734-749, doi:10.1109/TKDE.2005.99.

Azar, Yossi et al. "Spectral Analysis of Data." *Proceedings of the thirty-third annual ACM symposium on Theory of computing*, 2001, pp. 619-626.

Ballatore, Andrea et al. "Recomap: An Interactive and Adaptive Map-Based Recommender." *Proceedings of the 2010 ACM Symposium on Applied Computing*, 2010, pp. 887-891.

Becker, N. "Matrix Factorization for Movie Recommendations in Python." https://beckernick.github.io/matrix-factorization-recommender/. Accessed 9 February, 2020.

Beel, Joeran et al. "Paper Recommender Systems: A Literature Survey." *International Journal on Digital Libraries*, vol. 17, no. 4, 2016, pp. 305-338.

Bondarenko, K. "Precision and Recall in Recommender Systems", 2019. Available at: https://medium.com/@bond.kirill.alexandrovich/precision-and-recall-in-recommender-systems-and-some-metrics-stuff-ca2ad385c5f8. Accessed 11 February, 2020.

Canny, John. "Gap: A Factor Model for Discrete Data." *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, 2004, pp. 122-129.

Cho, Yoon Ho et al. "A Personalized Recommender System Based on Web Usage Mining and Decision Tree Induction." *Expert Systems With Applications*, vol. 23, no. 3, 2002, pp. 329-342.

Clement, J. "Global Retail E-Commerce Market Size 2014-2023. ." Statistica https://www.statista.com/statistics/379046/worldwide-retail-e-commerce-sales/. Accessed 21 July, 2020.

Cosley, Dan et al. "Is Seeing Believing? How Recommender System Interfaces Affect Users' Opinions." *Proceedings of the SIGCHI conference on Human factors in computing systems*, 2003, pp. 585-592.

Funk, Simons. "Gbolmier/Funk-Svd." https://github.com/gbolmier/funk-svd. Accessed 11 March, 2020.

Garcia, E. "Singular Value Decomposition (Svd) a Fast Track Tutorial", 2006. Available at: https://cs.fit.edu/~dmitra/SciComp/Resources/singular-value-decomposition-fast-track-tutorial.pdf. Accessed 14 March, 2020.

Gladwell, Malcolm. "The Science of the Sleeper." *BOOKSELLER*, 2000, pp. 22-25.

Goldberg, David et al. "Using Collaborative Filtering to Weave an Information Tapestry." *Communications of the ACM*, vol. 35, no. 12, 1992, pp. 61-70.

Golub, Gene H and Christian Reinsch. "Singular Value Decomposition and Least Squares Solutions." *Linear Algebra*, Springer, 1971, pp. 134-151.

Harper, F Maxwell and Joseph A Konstan. "The Movielens Datasets: History and Context." *ACM Transactions on Interactive Intelligent Systems (TiiS)*, vol. 5, no. 4, 2015, pp. 1-19.

Hofmann, Thomas. "Latent Semantic Models for Collaborative Filtering." *ACM Transactions on Information Systems (TOIS)*, vol. 22, no. 1, 2004, pp. 89-115.

Hug, Nicholas. "Surprise Python Scikit for Recommender Systems", 2015, Available at: http://surpriselib.com/. Accessed 12 April, 2020.

Kim, Eunju et al. "Purchase Propensity Prediction of Ec Customer by Combining Multiple Classifier Based on Ga." *International Conference on Electronic Commerce*, vol. 2000, 2000, pp. 274-280.

Koren, Yehuda et al. "Matrix Factorization Techniques for Recommender Systems." *Computer*, vol. 42, no. 8, 2009, pp. 30-37.

Marlin, Benjamin and Richard S Zemel. "The Multiple Multiplicative Factor Model for Collaborative Filtering." *Proceedings of the twenty-first international conference on Machine learning*, 2004, p. 73.

Min, Fan and William Zhu. "Mining Significant Granular Association Rules for Diverse Recommendation." *International Conference on Rough Sets and Current Trends in Computing*, Springer, 2014, pp. 120-127.

"Movielens." Group Lens Research https://movielens.org/. Accessed 3 June, 2020.

Peralta, Verónika. "Extraction and Integration of Movielens and Imdb Data." *Laboratoire Prisme, Université de Versailles, Versailles, France*, 2007.

Ponnam, Lakshmi Tharun et al. "Movie Recommender System Using Item Based Collaborative Filtering Technique." *2016 International Conference on Emerging Trends in Engineering, Technology and Science (ICETETS)*, IEEE, 2016, pp. 1-5.

Ricci, Francesco et al. *Introduction to Recommender Systems Handbook.* Springer, 2011.

Sarwar, Badrul et al. "Application of Dimensionality Reduction in Recommender System-a Case Study." Minnesota Univ Minneapolis Dept of Computer Science, 2000.

Schafer, B et al. "E-Commerce Recommendation Applications, Data Mining and Knowledge Discovery, Vol 5 (1–2)." Kluwer Academic Publishers, Boston, 2001.

Schafer, J Ben et al. "Collaborative Filtering Recommender Systems." *The Adaptive Web*, Springer, 2007, pp. 291-324.

Shardanand, Upendra and Pattie Maes. "Social Information Filtering: Algorithms for Automating “Word of Mouth”." *Proceedings of the SIGCHI conference on Human factors in computing systems*, 1995, pp. 210-217.

Yeung, Albert Au. "Matrix Factorization: A Simple Tutorial and Implementation in Python." 2010.

Zheng, Hua et al. "Do Clicks Measure Recommendation Relevancy?: An Empirical User Study." ACM, 2010 2010, pp. 249-252. Available at: doi:10.1145/1864708.1864759.