A comprehensive review of recommendation systems: method, data, evaluation and coding

João Gabriel de Moraes Souza[1, 7], Daniel O. Cajueiro[2,6,7], Johnathan de O. Milagres[4, 7], Vinícius Watanabe[2, 7], Vítor Bandeira Borges[2, 7], and Victor Rafael Celestino[4, 7]

[1]Business and Economics Department, Brazilian Institute of Capital Markets (Ibmec), Brazil.
[2]Economics Department, FACE, University of Brasília (UnB), Brazil.
[3]IFMG
[4]Business Department, FACE, University of Brasília (UnB), Brazil.
[6]Nacional Institute of Science and Technology for Complex Systems (INCT-SC). University of Brasília (UnB), Brazil.
[7]LAMFO, FACE, University of Brasília (UnB), Brazil.

October 6, 2023

Abstract

Contents

1 Introduction

A Recommendation System (RS) is an automatic system that provides suggestions of items that are likely to be handy for a particular individual. Recommendation System can be formulated in two different ways. The first is the Prediction problem, the objective is to predict the rating that a given user associates with a given item. The second is the Ranking problem, the objective is to rank the top items for a particular user. Note that if we have the solution for the former case, we naturally have the solution for the later. However, in some cases, we do not need to solve the first in order to solve the second.

A computational recommender systems appear in response to people's difficulty in choosing between a wide variety of products and services and among the various alternatives that are presented to them. The evolution of these systems and the fact that they work with large information bases allowed emerging (non-trivial) recommendations to be reached, providing even greater credibility than a human recommendation (Resnick & Varian, 1997).

These types of recommendations are used by various large-name companies like Google, Instagram, Spotify, Amazon, Reddit, and Netflix, often to increase engagement with users and the platform. For example, Spotify would recommend songs similar to the ones you've repeatedly listened to or liked so that you can continue using their platform to listen to music. Amazon uses recommendations to suggest products to various users based on the data they have collected for that user. Netflix uses recommendations to suggestion movies and series to various stereotypes of users based on the demographic and ratings collected for the users (Vatsal, 2021).

According to Schafer et al. (1999), many of the largest commerce Web sites are already using recommender systems to help their customers find products to purchase. A recommender system learns from a customer and recommends products that she will find most valuable from among the available products Fundamental to avoid information overload. A recommender system can be the crucial form to automate a constumation mass for E-commerce sites.

One of the economic incentives generated by the recommender system is free-ride by consuming evaluations. Future systems will likely need to offer some incentive for the provision of recommendations by making it a prerequisite for receiving recommendations or by offering monetary compensation. The second economic incentive is if someone can provide recommendations, content owners can generate mountains of positive recommendations for their own materials and negative recommendations for their competitors. Recommender systems also raise concerns about personal privacy. In general, the more information individuals have about recommendations, the better they will be able to evaluate those recommendations (Resnick & Varian, 1997).

The first Recommendation System is due to Rich (1979). She proposed proposed a Recommendation System using stereotypes (or clusters of characteristics) that plays a librarian and recommend relevant books to each user. She called this Recommendation System as Grundy.

It is worth mentioning that there are some other very interesting reviews about the theme of Recommendation System such as Adomavicius & Tuzhilin (2005), Su & Khoshgoftaar (2009), Koren et al. (2009) and Melville & Sindhwani (2010). Our attempt to provide this literature review is to allow the interest reader to understand the field of Recommendation System and the techniques used to do it without extensive background and after reading this review is able to know the main methods used, the datasets available to this task, the techniques used to evaluate Recommendation System and also how to code it.

Besides this introduction, we organize the manuscript as follows: Section 2 details the available datasets to the study of Recommendation System. Section 3 presents the basic topology of Recommendation Systems. Section 4 presents the main path traced by the literature for the construction and design of the main and most innovative recommendation algorithms used by the market and academia. Section 5 bring the most common evaluation methods of recommendation algorithms. Section 6 explores the most popular Recommendation Systems. Section 7

presents the main computational libraries used to solve recommendation algorithms. Finally, Section 9 brings the main conclusions of our paper.

2 Datasets

The Table 1 lists the most important datasets used for Recommendation Systems. Respectively it shows: First column, *Dataset name*, where a nickname for each dataset is displayed; Second column, *Number of Users*, that informs how many users, i.e. different IDs, were shown in the dataset; The third column, *Number of items*, informs how many items was available in the dataset; The fourth column, *Range*, informs how many rating possibilities the user had; The fifth column, *Domain*, tells which matter the dataset is about; The sixth column, *Type*, inform which type (Content-based, Collaborative filtering, etc.) the dataset is; The seventh column, *Density*, informs on average how many itens available each user, in the dataset, rated; The eighth, *Academic Reference*, shows the dataset source.

The bookcrossing is a website that connect readers around the globe, allowing them to rate, comment, exchange books, etc. Also, the bookcrossing have a important dataset of books, that is used for some RSs. This dataset is presented in (Ziegler et al., 2005), and count with 27,858 users, 271,379 items (books), which provided us 1,149,780 ratings (explicit/implicit).

GroupLens DataSet for movies, called MovieLens (ml) , it's one of the most famous datasets used for RSs, being referenced in many blogs and important RSs articles as (Resnick et al., 1994) and (Resnick & Varian, 1997). All the MovieLens datasets have a very similar structure, the main difference it is in the number of ratings, users and movies, but the ratings range still the same for all of them. The ml-25m dataset, for example, registered that: 162,541 users rated 59,047 movies 25 million times, in a range of 10 possible ratings (0.5 star - 5 stars). The others MovieLens datasets goes on the same idea, but with the different numbers.

Jester belongs to the list of important datasets, being hugely used among RSs, appearing in blogs and articles as (Gupta et al., 1999). The Jester dataset, is based on a rating of jokes. The dataset used in this article, has 73,421 users rating 100 jokes. Unlike the MovieLens dataset, the range of ratings on Jester goes from -10.00 to 10.00.

Table 1: A representative list of the most important datasets used for RSs.

Dataset name	Number of Users	Number of items	Range	Domain	Type	Density	Academic Reference
bookcrossing	278,858	271,379	11	Books	Content-based	0.0%	Ziegler et al. (2005)
filmtrust	1,508	2,071	8	Movies	Content-based	1.1%	Guo et al. (2013)
ml-25m	162,541	59,047	10	Movies	Collaborative filtering	0.2%	Resnick et al. (1994); Resnick & Varian (1997)
ml-20m	138,493	27,278	10	Movies	Collaborative filtering	0.5%	Resnick et al. (1994); Resnick & Varian (1997)
ml-10m	69,878	10,681	10	Movies	Collaborative filtering	1.3%	Resnick et al. (1994); Resnick & Varian (1997)
ml-1m	6,040	3,883	5	Movies	Collaborative filtering	4.2%	Resnick et al. (1994); Resnick & Varian (1997)
jester-ds	124,113	150	[-10,10]	Jokes	Collaborative filtering	31.5%	Gupta et al. (1999)

3 Basic topology of a Recommendation System

All Recommendation Systems depend on the following architecture:

1. Input interface for data collection: In many situations, we may represent the basic information necessary to build a recommendation system in a user-rating matrix, where each line refers to a user and each each column refers to the rating a given user values a given item. Depending on the RS, other pieces of information may be necessary, such as demographic information or other attributes of the user and attributes of the items.

2. Algorithm used to make suggestions: In general, the idea behind Recommendation Systems algorithms are very simple. They try to use efficiently the information about the tastes of the users in order to make new recommendations.

3. Output interface of recommendation: As we mention before, the interface of recommendation may come in different forms, depending on the objective of the system. The RS, for instance, may provide a sorted list of the most relevant items or a grade associated with each item.

4 Algorithms

In this section, we split and identified five different approaches used to make recommendations as well as M. J. Pazzani (1999). This division includes demographic filtering, collaborative filtering, content-based, knowledge-based, and hybrid.

4.1 Demographic Filtering

Demographic Recommendation System categorize the recommendations to the users based on their demographic attributes, as stereotypes, and their opinions about the items. Grundy Rich (1979) was the first and seminal Recommendation System. In the application in a Grundy, Rich (1979) plays the role of a librarian that collects personal information using an interactive system and a structured database to make a recommendation about books for users. In that same idea, M. J. Pazzani (1999), instead of having a structured database to learn from the users, they learn demographic information about the users using text classification.

User Modeling via Stereotypes

In the seminal work for Rich (1979), the author wants to write programs that distinctly treat their users based on the user's characteristics. The objective of the model is to generalize the stereotypes that are refined through inferences made by the program when interacting with the user.

As an example, in Rich (1979) paper, the Grundy system simulates an approximation of what a librarian would do when analyzing an individual's stereotype and, through this system, indicate the best book options for the possible tastes of that individual who has a generalizable stereotype, and therefore capable of receiving a recommendation with high accuracy. The inferred information about the user also allows the classification of the user into groups. These system builds a individual, inferred and long-term model of it's users, as the aiming to act as a librarian recommending books to the user.

If the system asks questions to the user for direct input is annoying and inefficient, since people are not good sources of information about themselves. Therefore, inferring all the details about a person is costly hence it requires a lot of interaction between the user and the program. Stereotypes are the solution to this dilemma as they allow the program to infer a lot of information about a user with little interaction creating clusters of stereotypes (Rich, 1979).

According to Rich (1979), in the foreground the system groups the main stereotypes, these groups are engendered by information directly provided by the users, with this information it is possible to infer about the users and provide them with the most appropriate recommendations. All inferences involve a degree of uncertainty - we do not know for sure if we are making a correct guess about the person's stereotypes. The stereotype tuple must also contain a value that indicates the degree of confidence, or the rating it gives to the certainty it has about its inference. The user synopsis may contain conflicting information about a certain facet based on different stereotypes a person belongs to. Thus it is necessary that the user synopsis also retains information on why it attributed a particular stereotype to a person.

The Grundy organizes the stereotypes it knows in a directed acyclical graph. There is thus an initial node that contains the most general stereotype. This stereotype has median values on all facets and low confidence for all of them. Particular nodes - stereotypes - may have information that overrides the general class to which they belong. By adjusting the Grundy system's grouping by stereotypes, the system periodically updates with new triggers, perhaps the initial idea of reinforcement learning.

The new triggers look at all existing stereotypes and their facets, if they are not already in the user synopsis, the system adjusts the new facets belonging to the new stereotype. After the modification of existing stereotypes is creating new ones. These could be done after a system had enough models of individual users to be able to abstract patterns from them. The construction of new stereotypes can be done using straightforward pattern classification techniques (Rich, 1979).

A framework for collaborative, content-based and demographic filtering

Intending to describe how each type of information may be used individually and then discussing an approach to combining recommendations from multiple sources, M. J. Pazzani (1999) developed the paper to illustrate each approach and combined these approaches in the context of recommending restaurants. The authors show different approaches within a common framework (M. J. Pazzani, 1999).

The paper combining recommendations from multiple approaches find and recommend information sources for an individual user that have been rated highly by other users who have a pattern of ratings similar to that of the user, and recommendations based upon demographic information (M. J. Pazzani, 1999). Attempt to compare this different approaches and techniques for combining recommendations. Using restaurant ratings given by 44 students at the University of California. The users all provided ratings for 58 Web pages that describe restaurants in Orange County, California. Approaches that combine multiple sources of information have higher precision than approaches based upon a single type of information.

According to M. J. Pazzani (1999), two basics approaches have emerged for making recommendations:

- Collaborative filtering - recommend information sources based upon demographic information highly rated by users with similar interests.
- Content-based filtering - analyzes content of information sources to create a profile of the user's interests.

In collaborative recommendations is an approach to making recommendations by finding correlations among users of recommendation system and using it as weights for predicting the rating that the current user would give to an item (M. J. Pazzani, 1999). Positive rating have vale $+1$ and negative rating -1. The Pearson Correlation Coefficient r in Equation 1 can be used in this circumstance. Let R_{ij} be the rating of user I on document j.

$$r(x,y) = \frac{\sum_{d \in documents}(R_{x,d} - \bar{R}_x)(R_{y,d} - \bar{R}_y)}{\sqrt{\sum_{d \in documents}(R_{x,d} - \bar{R}_x)^2 \sum_{d \in documents}(R_{y,d} - \bar{R}_y)}} \quad (1)$$

Where $\bar{R}_x$ is the mean value of ratings for user x.

In content-based methods that make recommendation by analyzing the description of the items that have been rated by the user and the description of items to be recommend (M. J. Pazzani, 1999). All of the content-based approaches represent documents by the " important " words in the documents. The informative words are found by finding the expected information gain, that the presence or absence of a word W gives toward the classification of elements of a set of documents S (M. J. Pazzani, 1999):

$$E(W,S) = I(S) - [p(W = present)I(S_{W=present}) + p(W = absent)I(S_{W=absent})] + p(W = absent)I(S_{W=absent}) \quad (2)$$

where

$$I(S) = \sum_c -p(S_c)log_2(p(S_c))$$

and $P(W = present)$ is the probability that W is present in a document, and $S_{W=present}$ is the set of documents that contain at least one occurrence of W and S_c is the documents that belong to class c. Once the representation has been in the documents, the classification of algorithms can learn a profile to distinguish representations of highly rated documents form others. Winnow algorithm Littlestone & Warmuth (1994); Blum (1997) learns the weight w_i associated with each word x_i to form a linear threshold function: $\sum w_i x_i > \tau$. By recursively adjusting the weights until all examples are processed correctly.

In Demographic information, the system is used to identify the types of the stereotype of users that like a certain object. Recommendation systems aim to predict users' interests and recommend items most likely to interest them. The paper of Yassine et al. (2021), propose a new intelligent recommended system that combines collaborative filtering with the popular unsupervised machine learning algorithm K-means clustering. Furthermore, the paper user demographic attributes such as the gender and age to create segmented user profiles, when items, in these case movies, are clustered by genre attributes using K-means and users are classified based on the preference of item sand the genres they prefer to watch.

Recommender systems provide personalized recommendations to users from an abundance number of possible options in online stores. Matrix factorization is a well-known and accurate collaborative filtering approach for recommender systems, which suffer from the cold-start problem for new users and items. For this collaborative filtering, this method combined demographic models and ratings of evaluation about determinated products Hashemi & Rahmati (2020).

Finally the Collaborative methods for similarities between users to make prediction. Ratings of individuals users is used to determine similarity M. J. Pazzani (1999).

A Framework for Recommender systems

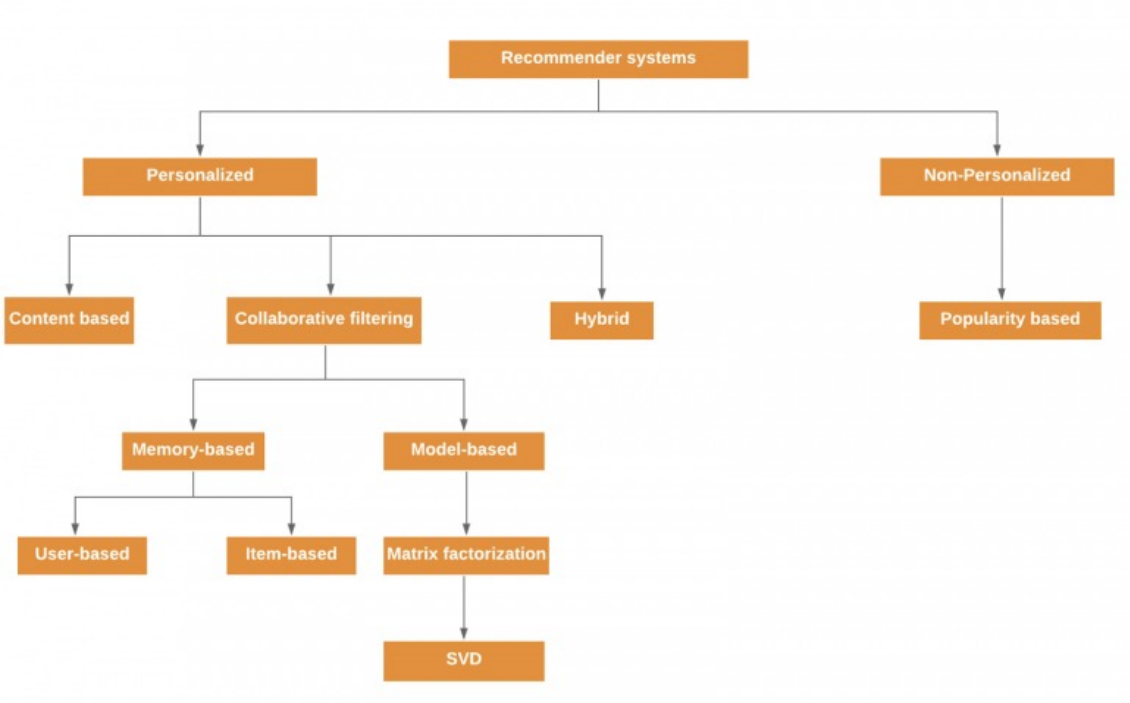

Figure 1: Recommender Systems Flowchart
Source: Thingssolver (n.d.)

Figure 1 illustrates, in a fluid way, how Recommender systems are divided. These systems can be divided into Non-Personalized and Personalized. Non-Personalized recommendation systems can be based on popularity. In this case, the system will recommend items that most users are liking, for example. On the other hand, Personalized systems use users' data to get them customized recommendations. They also have much more types and ramifications.

Further, in this section, specific models and cases are discussed. Recommender Systems and it's models evolved and changed over the years, leading to more complex methods, but more capable of overcoming challeges that appeared in earlier ones. Personalized models are more popular in literature and also have a significant background to work on (such as libraries and documentation).

- Content-based

  In Content-based recommendation systems, the way to create a custom recommendation in this category is based on the idea of recommending items to a certain user that are similar to something that the user liked in the past or is looking at in the present time.

- Collaborative-filtering

  Collaborative-filtering is based on the assumption, that you can divide people into groups by their past behavior, therefore they will have more alike tastes. There are two types of Collaborative-filtering, Memory-based e Model-based.

  (a) Memory-based

  Memory-based type uses past rating data to compute similarities between users or items. If we are interested in user similarities, we have a user-based model, where users that, for example, rated similar items will get their preferences used to recommend new items. If we are interested in item similarities, we will have an item-based

model. In this case, we have a given item, then the model finds a similar item to recommend to the user.

(b) Model-based

Those models provide recommendations by developing models from user ratings using machine learning algorithms. The most famous approach is Matrix Factorization. This approach works basically on the idea that, given feedback from a user about some product, we can transform this info into a matrix where the rows represent users and the columns represent an item. This model is used to solve the sparsity problem, by finding a set of latent factors and then determining users' preferences. One interesting subgroup of Matrix Factorization is the SVD model, which is a model that uses the user-item matrix to extract some features and correlations.

- Hybrid

  Ultimately, we have the Hybrid system. This one is a combination of different approaches. Nowadays most recommender systems use the hybrid approach. The idea is that a combination of different ways to attend to a problem can give us better results.

The social implications of future systems will likely need to offer some incentive for the provision of recommendations by making it a prerequisite for receiving recommendations or by offering monetary compensation. If anyone can provide recommendations, content owners may provide mountains of positive recommendations for their materials and negative recommendations for their competitors. Recommender systems also raise concerns about personal privacy.

The first Business Models can charge recipients of recommendations, in another way a second model for cost recovery is advertiser support. Presumably, advertisers would find such systems very useful, since they generate detailed marketing information about consumers. A third model is to charge a fee to the owners of the items being evaluated.

4.2 Collaborative-filtering

Goldberg et al. (1992) coined the term "collaborative-filtering" in the context of the first commercial RS. They designed this RS to recommend documents drawn from newsgroups to a collection of users. In collaborative-filtering, we use the known cells of the user-rating matrix to predict the missing cells. These methods remind the methods used to recover cells of tables that present missing data (Little & Rubin, 2019).

We may split the collaborative-filtering algorithms in two groups, namely memory-based and model-based algorithms. The explicit difference between these two types of methods is that while the former uses non-parametric models, the later uses parametric ones.

4.2.1 Memory-based

Memory-based methods use non-parametric models to recover the missing information of the user-rating matrix. These methods belong to the class of *instance learning based* methods, such as the supervised K-Nearest Neighbors, where the prediction is the dependent on the object that is being predict (Park et al., 2015). We may split these methods in two groups:

1. User-item collaboration: We find the users that have a similar behavior to a target user. We use information about the ratings of the similar users to predict the ratings of these items for the target user.

   Recommendation systems are a useful help to the users and a very important tool to business, like E-commerce sales. But one of the challenges is to improve the quality of the recommendations, because users need a trustworthy recommendation. One way to

achieve a high level of quality, is by taking a user interaction with a product, then use this information to reach a other user with similar taste (B. Sarwar et al., 2001; Greening, 1997; Shardanand & Maes, 1995).

Greening (1997) presented us that the trust of the user is fundamental in consumer decisions. Therefore, the method presented on the article uses the idea that "consumer preferences are not random", so people can recommend products to each other, based on they ratings. With that principle they were able to present a method with a high accuracy of recommendation.

2. Item-item collaboration: We determine the items that are similar to a given target item that we need information. Then, we use the ratings of the similar items to predict the target item.

   The Item-based techniques emerges as a way to address the scalability problems of User-based methods. Some authors suggests that item-based techniques, have a better performance and quality than the user-based methods (B. Sarwar et al., 2001; B. M. Sarwar et al., 1998; Deshpande & Karypis, 2004).

   One Item-based technique is called "Clustering". This method, basically divide the item data in groups, then classifies the groups by their similarity. The cluster models are able to compare the user with an limited number of segments, unlike other ones. Therefore, the results are way better on scalability and performance (Linden et al., 2003; O'Connor & Herlocker, 1999; Ungar & Foster, 1998).

   Another form of Item-based techniques, is the Graph based approach. Aggarwal et al. (1999), introduces us into an algorithm, that proves to be a fast, scalable, accurate and have a small learning curve. This method basically give us the rating of a certain item for a certain user, by computing the item as a weight average.

Note that in both cases, we have two steps. In the first step, we determine the similar cells in the user-rating matrix we will use. In the second step, we use the information (rating) of these cells to predict the missing cells.

Resnick et al. (1994) present the classical approach to user-item collaboration filtering. In the first step, they determine the similarity among the target user and all the other users of the system using the Pearson correlation coefficient. In order to evaluate these correlations they use the ratings provided by the target user and the ratings provided for each other user of the system. Then, in the second step, it evaluates the ranking of the item j for the target user i using

$$r_{ij} = \overline{r}_i + \sum_{k \in \mathcal{K}} \frac{(r_{kj} - \overline{r}_k)\rho_{ik}}{\sum_{k \in \mathcal{K}} |\rho_{ik}|}, \tag{3}$$

where $\overline{r}_i$ is the average ranking of user i and $\rho_{ik}|$ is the Pearson correlation between user i and user k.

4.2.2 Model-based

Different from the memory based methods, model-based methods use machine learning supervised models to find out whether we should or not recommend an item to an user. This is typically a classification problem and we can use any supervised model to do it. In the first step, we use the known cells to build the model. In the second step, we use the learned model either to predict the rating of the item or to predict whether we should or not recommend an item. Since we are interested in specific models for each user, We usually associate a classifier with each user. In this case, for a target user, we can assume that each other user is a different feature that can be used to predict the ratings of the target user.

The seminal work that puts this idea explicitly into practice is Billsus et al. (1998). It defines two classes, *like* and *dislike*, and use them as class labels. It tests different supervised models such as neural network models and reduces the dimension of the problem using singular value decomposition. Another classical contribution to this field is due to Miyahara & Pazzani (2000). It also defines the *like* and *dislike* classes like in in Billsus et al. (1998). In order to classify the items, it uses the naive Bayes model. It selects the features using the expected information gain (Quinlan, 1986).

As presented before, some collaborative filtering methods have some problems and limitations. One way to address this issue, is a model-based solution presented on Billsus et al. (1998). Basically we must take an approach on collaborative filtering as a classification task, and therefore applying classification algorithms.

One interesting model-based method, is the association rules. To comprehend this idea, let's think on a supermarket, where hundreds of products are sold every day. Each client purchase is registered, thus creating a database. The association rule model, basically takes all this data of purchases, and then correlate each product. For example: 95% of transactions that purchase shampoo and hair cream, also purchase soap (Agrawal et al., 1993, 1994).

But those databases are very huge, so we can have some problems of correlating some items with different frequencies, therefore generating some insignificant information. To solve this kind of problem it's proposed to adjust the minimum support, which is a part that controls the minimum number of data cases that a rule must cover (Lin et al., 2002; Liu et al., 1999).

In collaborative filtering, we have some solutions that are less complicated than the others, and still very efficient. One of those are the Simple Bayesian Classifier (M. Pazzani & Billsus, 1997). This algorithm works on the assumption that a feature presence is not related to the presence of another feature.

A variant of this model, is the Sparsed Data Model. This model assumes that known features are the only ones that can be used as a informative feature for classification. As proven in Miyahara & Pazzani (2000) paper, this model performs better than correlation-based models, which are the a typical type of model used in collaboration filtering.

Another model for collaborative filtering, is the Probabilistic Relational Model. This method, basically, assumes that, in a set of objects, the relation between them are fixed. Therefore, the Probabilistic Relational model will define only a probability distribution over the attributes of the objects in the model Getoor et al. (2007).

Probabilistic Relational Model can be very useful, once that is possible to have a PRM identifying classes of users by all collected information, for example the propensity to visit some web pages, and the user purchase behavior. Also, the PRM can create a recommendation for new users, just by demographic informations, while standard methods of collaborative filtering can't Getoor et al. (1999).

The Support Vector Machines (SVM), is a very successful method in text categorization, therefore we may assume that it will be efficient as a recommender system. But when it comes to recommendation systems, SVM presents some issues due to the sparsity problem.

One solution to enable an application of this model on recommender systems, is a SVM-based heuristic method, as presented by (Xia et al., 2006). Thus, the missing values are repeatedly estimated in the user-item matrix.

Neural Networks are widely used in learning tasks. This class of algorithms is, in a few words, learning algorithms that store information and works in a interconnected way, similar to the brain neurons. Therefore, this algorithms can be applied in collaborative filtering problems as presented in Ziegler (2006).

A type of neural network is the Restricted Boltzmann Machines (RBM). It's known that some of the collaborative filtering approaches cannot deal in an effective way with a huge dataset. The RBM models, are able to solve this kind of problem by using a class of two-layer undirected graphical models. This model can be applied in a dataset containing over 100 milion

users/movie ratings, and still be successfully (Salakhutdinov et al., 2007).

One interesting idea for collaborative filtering models is the application of neural embedding algorithms. In Neural Language Processing, those algorithms provide representations of words as vectors, for example. On his paper,(Barkan & Koenigstein, 2016) introduce us to the possibility of apply those models to collaborative filtering situations, by learning item similarities by embedding items in a low dimensional space.

Another neural related models, is called deep neural networks. One of the ways to explore this area, is presented on He et al. (2017), where it's put an effort to develop techniques of neural networks, to approach collaborative filtering problems by using implicit feedback.

Collaborative filtering system, although very successful, has some problems. One of them is that, in some models, the performance of the system degrates the higher the number of customer and products. One idea to solve this issue, is using a matrix factorization technique, called Singular Value Decomposition (SVD), to reduce the dimensionality of the database (B. Sarwar et al., 2000).

There are also, some works that focus on different forms of factorization in recommendation systems, with the intent of create a more efficient algorithm. For example, Srebro et al. (2004), shows us an approach that uses low-norm factorization, while S. Zhang et al. (2006) uses a low-dimensional linear model.

We have some approaches that uses the SVD idea with some models, like the Principal Component Analysis (PCA) explored on Kim & Yum (2005), where the PCA is employed with SVD for estimating missing ratings and to reduce the dimension. Also we have Brand (2003), where it is used a "thin" SVD with some revision rules.

One interesting approach is presented in Canny (2002), where it is used a probabilistic factor analysis model. But we also have some models like the Hofmann (2004), where it is offered a model that can build statistical techniques for inference and model selection, or even more complexes approaches such as Rennie & Srebro (2005), that uses gradient-based optimization methods.

The Maximum Margin Matrix Factorization (MMMF), addressed on Rennie & Srebro (2005), was a very important and wisely explored, such that later articles have applied this idea with some different approaches. For example DeCoste (2006), proposes that ensembles of MMMF's can give stronger predictions than a single MMMF. Also Weimer et al. (2007) is based on this idea, but the algorithm optimizes ranking measures, not rating measures.

Finally we have some other kind of approaches, such as regression models. One of those are Vucetic & Obradovic (2005), which deals with the sparsity problem. In this method, is explored an model that learns a number of experts describing relationships in ratings between pairs of items, where those experts are, lately, combined to predict the preference of the remaining items. We also have Mild & Natter (2002), where linear regression is tested to deal with a large dataset. Here we found that in a large number of customer, simple linear regression with model selection, can be better than other types of regression. Another example is presented on (Bell & Koren, 2007), where it is developed an algorithm that enhance prediction accuracy of neighborhood-based models, through a linear regression.

Other works also use linear models to present something new Lemire & Maclachlan (2005) uses a linear form that, pre-computes the average difference between ratings of a item and another one for users that rated both. On the other hand we have some combine methods, like a neighborhood method based on formally optimizing a global cost function, combined with some extensions of SVD-based latent factor models, which can improve the accuracy (Koren, 2008). Also is important to emphasize, that linear models are more accurate than most of memory-based methods, as proven in T. Zhang & Iyengar (2002).

4.3 Content-based

Content-based RSs find similar items to the items that the users liked in the past using the attributes of the liked items and the attributes of the unknown items. The focus of these systems is on the attributes that the users liked in the past.

The early content-based RS is the so-called Syskill & Webert (M. J. Pazzani et al., 1996), which is an intelligent agent designed to learn website profiles. It converts the HTML source of a web page into a Boolean feature vector. Each feature has a Boolean value that indicates whether a particular word is present (at least once) or absent in a particular web page. The choice of the words that it uses as features is based on an information-based approach (Quinlan, 1986).

In M. J. Pazzani & Billsus (2007), this approach is compared with more sophisticated and computationally expensive methods. The results shows that Bayesian Classifier performs at least as well as the other methods. ID3 for Quinlan (1986) was not accurate in this task, and was outperformed by M. Pazzani & Billsus (1997) new approach (which is a improvement of Syskill & Webert).

Considering that user preferences change over time, so do their behaviors. For that reason, sequential recommendations became a prevalent topic over the years in RSs literature. Channarong et al. (2022) cites BERT4Rec model which uses the

bidirectional-encoder-representations-from-transformers (BERT) to model sequential behavior by historical data, configuring a content-based filtering approach. The author argues that taking only historical data into account weakens model's result. Therefore, a new method called HybridBERT4Rec is proposed, it is a union from both content-based filtering and collaborative filtering approaches. HybridBERT4Rec generates two profiles that can be used for rating scores, with a higher accuracy than BERT4Rec.

Balabanović (1997) also points the idea that individual preferences changes over time, and call the models which deal with this kind of problem as adaptive recommendation services. The paper propose a model called "Fab" adaptive Web page recommendation service. Fab also combines shared interest among users and representations provided by content analysis.

Online recommendation are divide in three-stage process: Collection; Selection; and Delivery. Fab comes as a method to collect and select web pages. It stables a middle ground between content-based and collaborative. Balabanović & Shoham (1997) shows how Fab can eliminate the problems found in each approach. The paper's experiment showed that Fab outperforms other models.

4.4 Knowledge-based

Both collaborative-filtering-based and content-based systems need a relevant amount of data to perform well. The lack of data is a typical problem that arises with new users. This difficulty in the literature is called the *the cold-start problem*. Besides happening with new users of a system, it is also common in situations where the items to be suggested are not consumed frequently such as cars, houses and luxury items.

Knowledge-based systems are split in two groups:

1. Constrained-based: Users constrain the caracteristics of the item they wish.

2. Case-based: Users provide instances of items as a reference.

Unlike other problem-solving tasks, a comprehensive formalization for Knowledge-based recommender systems was not yet available. Recently, Cena et al. (2021) provided a logical foundation in their attempt to unify different approaches. The possibility of comparing systems based on *the way they use knowledge to generate recommendations* typically does not attract

enough attention in the literature, but the possibility of choosing among different recommendation generation strategies based on their underlying semantics can provide very interesting insights on the task itself.

Based on the division made by Cena et al. (2021), Knowledge-base recommender system can be divided in 5 families: context-aware recommenders; contraint-based recommenders; negative-preferences based recommendations; package recommendations; and Group recommendations.

Context-aware recommenders depend on the user preferences and contextual information. Different definitions of context in the literature had been outlined. Schilit & Theimer (1994), Brown et al. (1997), Adomavicius et al. (2005), and even Cena et al. (2021) defines context, each one in a different and unique way - although with the same essence.

Contraint-based recommenders use constraints as a formalism to represent compatibilities (Felfernig & Burke, 2008). In this kind of family, users explicitly inform their preferences by informing characteristics of the item they wish. Negative-preferences based take into account what users don't like for the model. It can be used to inform which items the user should avoid (Y. Zhang et al., 2014). Package recommendations consists in the recommendation of items that tends to be consumed together as pointed out by Benouaret & Lenne (2016). Group recommendations involves the resolution of aggregation problem. Some examples are: maximizing group pleasure; avoid that a member is dissatisfied, etc. (Masthoff, 2004).

4.5 Hybrid

To create a hybrid content-based or collaborative system, the system maintain user profiles based on content analysis, and directly compare these profiles to determine similar users for collaborative recommendation. Users receive items both when they score highly against their own profile, and when they are rated highly by a user with a similar profile.

Table 2 shows important articles cited along section 4. Article's Author(s) were ordered by Alphabetical Order. Among this articles, different kinds of algorithm were exposed and they were classified in the second column *Algorithm*.

Table 2: Literature review in algorithms

Article's Author(s)	**Algorithm**
Aggarwal et al. (1999)	Collaborative Filtering; Memory-based
Agrawal et al. (1993)	Collaborative Filtering; Model-based
Agrawal et al. (1994)	Collaborative Filtering; Model-based
Balabanović (1997)	Content-based
Balabanović & Shoham (1997)	Content-based
Bell & Koren (2007)	Collaborative Filtering; Model-based
Billsus et al. (1998)	Collaborative Filtering; Model-based
Brand (2003)	Collaborative Filtering; Model-based
Canny (2002)	Collaborative Filtering; Model-based
Cena et al. (2021)	Knowledge-based
Channarong et al. (2022)	Content-based
DeCoste (2006)	Collaborative Filtering; Model-based
Deshpande & Karypis (2004)	Collaborative Filtering; Model-based
Getoor et al. (1999)	Collaborative Filtering; Model-based
Getoor et al. (2007)	Collaborative Filtering; Model-based
Goldberg et al. (1992)	Collaborative Filtering
Greening (1997)	Collaborative Filtering; Model-based
He et al. (2017)	Collaborative Filtering; Model-based

Hashemi & Rahmati (2020)	Collaborative Filtering; Model-based
Hofmann (2004)	Collaborative Filtering; Model-based
Kim & Yum (2005)	Collaborative Filtering; Model-based
Koren (2008)	Collaborative Filtering; Model-based
Lemire & Maclachlan (2005)	Collaborative Filtering; Model-based
Lin et al. (2002)	Collaborative Filtering; Model-based
Linden et al. (2003)	Collaborative Filtering; Memory-based
Little & Rubin (2019)	Collaborative Filtering
Liu et al. (1999)	Collaborative Filtering; Model-based
Mild & Natter (2002)	Collaborative Filtering; Model-based
Miyahara & Pazzani (2000)	Collaborative Filtering; Model-based
O'Connor & Herlocker (1999)	Collaborative Filtering; Memory-based
Park et al. (2015)	Collaborative Filtering; Memory-based
M. J. Pazzani (1999)	Content-Based; Demographic Filtering
M. J. Pazzani et al. (1996)	Content-Based
M. Pazzani & Billsus (1997)	Content-Based
M. J. Pazzani & Billsus (2007)	Content-Based
Quinlan (1986)	Collaborative Filtering; Model-based
Rennie & Srebro (2005)	Collaborative Filtering; Model-based
Resnick et al. (1994)	Collaborative Filtering; Memory-based
Rich (1979)	Collaborative Filtering; Demographic Filtering
Salakhutdinov et al. (2007)	Collaborative Filtering; Model-based
B. Sarwar et al. (2000)	Collaborative Filtering; Model-based
B. Sarwar et al. (2001)	Collaborative Filtering; Memory-based
Shardanand & Maes (1995)	Collaborative Filtering; Memory-based
Srebro et al. (2004)	Collaborative Filtering; Model-based
Ungar & Foster (1998)	Collaborative Filtering; Memory-based
Vucetic & Obradovic (2005)	Collaborative Filtering; Model-based
Weimer et al. (2007)	Collaborative Filtering; Model-based
Xia et al. (2006)	Collaborative Filtering; Model-based
Yassine et al. (2021)	Collaborative Filtering; Model-based
T. Zhang & Iyengar (2002)	Collaborative Filtering; Model-based
S. Zhang et al. (2006)	Collaborative Filtering; Model-based
Ziegler (2006)	Collaborative Filtering; Model-based

5 Evaluation methods

The literature divides the methods for evaluating the quality and effectiveness of the RS algorithms. The metrics here presented are focused on comparing different personalized collaborative-filtering models.

Some of the most used metrics in Artificial Intelligence and Statistics, that focus on being proportional to the size of errors, are frequently used in evaluations. Let's first define some notation (this is the same notation used in Koren & Sill (2013)) that is useful when mathematically expressing these formulas.

We are given ratings for m users and n items, with u, v indexing users and i, j indexing items. In addition, we index rating values by r. A rating r_{ui} indicates the numerical rating which the user u gave to an item i, where high values mean stronger preference. We distinguish

predicted ratings from known ones, by using the notation $\hat{r}_{ui}$ for the predicted value of r_{ui}. The set of items rated by user u (in the train set) is denoted by $R(u)$. The overall training set, containing all rated user-item pairs $(u, i, r = r_{ui})$ is denoted by $\mathcal{R}$. The test set is denoted by $\hat{\mathcal{R}}$.

There are three main metrics that are proportional to the size of prediction errors. The first two use a quadratic approach, the third uses the module operator. MSE (Mean Square Error) is the mean of quadratic errors, and can be expressed as:

$$MSE(\hat{\mathcal{R}}) = \frac{1}{|\hat{\mathcal{R}}|} \sum_{\hat{r}_{ui} \in \hat{\mathcal{R}}} (r_{ui} - \hat{r}_{ui})^2 \tag{4}$$

The RMSE (Root Mean Square Error) is equal to the square root of the MSE.

$$RMSE(\hat{\mathcal{R}}) = \sqrt{\frac{1}{|\hat{\mathcal{R}}|} \sum_{\hat{r}_{ui} \in \hat{\mathcal{R}}} (r_{ui} - \hat{r}_{ui})^2} \tag{5}$$

The MAE (Mean Absolute Error) uses the module operator instead of the quadratic.

$$MAE(\hat{\mathcal{R}}) = \frac{1}{|\hat{\mathcal{R}}|} \sum_{\hat{r}_{ui} \in \hat{\mathcal{R}}} |r_{ui} - \hat{r}_{ui}| \tag{6}$$

Koren & Sill (2013) formulated a new metric for evaluations based in ordinal preferences. The previous metrics have some issues, they assume numerical rating values. They cannot express rating scales, which vary among different users. Second, it cannot be applied in cases where ratings are ordinal. Thus, besides using RMSE, we also employ a ranking-oriented metric which is free of the aforementioned issues.

Given a test set $\hat{\mathcal{R}}$, we define the number of concordant pairs for user u by counting those ranked correctly by rating predictor $\hat{r}_u$·

$$n_c^u = |\{(i, j) \mid \hat{r}_{ui} > \hat{r}_{uj} \text{ and } r_{ui} > r_{uj}\}| \tag{7}$$

We count the discordant pairs n_d^u for user u in a similar fashion.

Summing over all users we define $n_c = \sum_u n_c^u$ and $n_d = \sum_u n_d^u$. The quality metric we use measures the proportion of well ranked items pairs, denoted by FCP (for Fraction of Concordant Pairs)

$$FCP(\hat{\mathcal{R}}) = \frac{n_c}{n_c + n_d} \tag{8}$$

a measure that generalizes the known AUC metric into nonbinary ordered outcomes.

6 Popular Recommendation Systems

One interesting recommendation system, it's the PHOAKS system, referenced in Terveen et al. (1997). PHOAKS stands for People Helping One Another Know Stuff, and has an very innovative proposal for it's time. Basically the system uses exchanged messages between users from "newsgroups" of the Usenet(A communication system precursor to the internet forums), and automatically recognize, tallies and redistribute recommendations from the Usenet.

The article written by Kautz et al. (1997), demonstrates that social networks (in this case it's a group of people linked by professional activities) works in a idea that any two individuals, in terms of direct personal relationships are relatively close to each other. Concurrently, people have a limit to the information they are willing to make public, which can makes a search for some information or a referral chain, a costly task. Thus, the Referral Web system, as proposed on the article, is a solution for this problem. The system rebuild, visualize and search social networks in the World-Wide Web. The system, basically uses the social network to make an search like any other generic search engine, but with a well defined focus, which provides the users awareness of their communities.

Examples of such systems are InfoFinder, which is an system that learns the user preferences from online documents the user have classified.(Krulwich, 1996). Another one is the NewsWeeder, that addreses the problem of creation an maintenance of the user profile, by allowing the uses rate their interest level for certain articles, and then learning a profile based on those ratings. (Lang, 1995).

7 Libraries and commercial softwares

There are now several libraries with implementations of the most common methods of RSs. Common RSs implementations tend to use open-source packages, as it's quality is robust enough and allows changes depending on the operator's will.

In that sense, Surprise, an open-source Pythong package, is an abbreviation for *Simple Python Recommendation System Engine*. Since the focus here is on the simplicity, the package includes all functions necessary for the full implementation of a Recommendation System. One strong point about Suprise is the documentation. Surprise documentation presents a robust and detailed explanation of how the package works and it's functionalities.

Another library worth mentioning is TensorFlow Recommenders. This package contains dataset examples and tools like Surprise for the Recommendation System implementation. TensorFlow Recommenders documentation is focused on beginners and professionals, being detailed enough for different purposes.

Recmetrics is a package that focuses on the metrics. This package is useful for evaluating the recommendation metrics, containing in example: Long Tail Plot; Coverage; Novelty ;Personalization, etc. Some commercial softwares for Recommender System are: Amazon Machine Learning, that works as a machine learning plataform letting the user model data and create predictions (*Amazon Machine Learning*, n.d.); and IBM Watson, it is used to solve complex problems (*IBM Watson*, n.d.) . Benefits from the use of commercial softwares include: the possibility to have in hand a robust model without the necessity to make a big investment from the start. Also, they have continuous update and improvements, making the process easier for costumers to implement such recommend systems.

8 Results

Model Description

The surprise library provides 11 classifier models that try to predict the classification of training data based on several different collaborative-filtering techniques. The models provided with a brief explanation in English are mentioned below, for more information please refer to the package documentation.

random_pred.NormalPredictor: Algorithm predicting a random rating based on the distribution of the training set, which is assumed to be normal. baseline_only.BaselineOnly: Algorithm predicting the baseline estimate for given user and item.

knns.KNNBasic: A basic collaborative filtering algorithm.

knns.KNNWithMeans: A basic collaborative filtering algorithm, taking into account the mean ratings of each user.

knns.KNNWithZScore: A basic collaborative filtering algorithm, taking into account the z-score normalization of each user.

knns.KNNBaseline: A basic collaborative filtering algorithm taking into account a baseline rating.

matrix_factorization.SVD: The famous SVD algorithm, as popularized by Simon Funk during the Netflix Prize. matrix_factorization.SVDpp: The SVD++ algorithm, an extension of SVD taking into account implicit ratings.

matrix_factorization.NMF: A collaborative filtering algorithm based on Non-negative Matrix Factorization. slope_one.SlopeOne: A simple yet accurate collaborative filtering algorithm.

co_clustering.CoClustering: A collaborative filtering algorithm based on co-clustering. It is possible to pass a custom dataframe as an argument to this class. The dataframe in question needs to have 3 columns with the following name: ['userID', 'itemID', 'rating'].

Limitations and bias

In this model we have faced some obstacles that we had overcome, but some of those, by the nature of the project, couldn't be totally solved. Databases containing profiles of possible users of the planned prototype are not available. For this reason, it was necessary to carry out simulations in order to represent the interests of these users, so that the recommendation system could be modeled. A simulation of clusters of latent interests was realized, based on topics present in the texts describing financial products. Due the fact that the dataset was build it by ourselves, there was no interaction yet between a user and the dataset, therefore we don't have realistic ratings, making the results less believable.

Later on, we have used a database of scrappings of linkedin profiles. The problem is that the profiles that linkedin shows is biased, so the profiles that appears was geographically closed, or related to the users organization and email.

Training data

To train the Latent Dirichlet allocation (LDA) model, it was used a database of a scrapping of Researchers profiles on Linkedin.
The avaliable database are:

- ml_100k
- ml_1m
- jester
- lda_topics
- lda_rankings
- uniform

The avaliable methods are:

- surprise.NormalPredictor
- surprise.BaselineOnly
- surprise.KNNBasic
- surprise.KNNWithMeans
- surprise.KNNWithZScore
- surprise.KNNBaseline
- surprise.SVD
- surprise.SVDpp
- surprise.NMF
- surprise.SlopeOne
- surprise.CoClustering

The avaliable evaluators are:

- surprise.rmse
- surprise.mse
- surprise.mae
- surprise.fcp

MSE: 0.9146

Table 3: LDA-GENERATED DATASET- ranking

	RMSE	MSE	MAE	FCP
NormalPredictor	1.820737	3.315084	1.475522	0.514134
BaselineOnly	1.072843	1.150992	0.890233	0.556560
KNNBasic	1.232248	1.518436	0.936799	0.648604
KNNWithMeans	1.124166	1.263750	0.808329	0.597148
KNNWithZScore	1.056550	1.116299	0.750004	0.669651
KNNBaseline	1.134660	1.287454	0.825161	0.614270
SVD	0.977468	0.955444	0.757485	0.723829
SVDpp	0.843065	0.710758	0.670516	0.671737
NMF	1.122684	1.260420	0.722101	0.688728
SlopeOne	1.073552	1.152514	0.747142	0.651937
CoClustering	1.293383	1.672838	1.007951	0.494174

Table 4: BENCHMARK DATASET- uniform

	RMSE	MSE	MAE	FCP
NormalPredictor	1.508925	2.276854	1.226758	0.503723
BaselineOnly	1.153331	1.330172	1.022732	0.506818
KNNBasic	1.205058	1.452165	1.026591	0.501168
KNNWithMeans	1.202024	1.444862	1.028149	0.503527
KNNWithZScore	1.216041	1.478756	1.041070	0.501582
KNNBaseline	1.225609	1.502117	1.048107	0.498198
SVD	1.176273	1.383619	1.013285	0.502067
SVDpp	1.192619	1.422340	1.018717	0.500909
NMF	1.338216	1.790821	1.120604	0.492944
SlopeOne	1.224219	1.498713	1.047170	0.494298
CoClustering	1.223020	1.495778	1.033699	0.518509

Benchmarks

9 Final Remarks

On recommendation systems, we have some challenges to overcome as presented by the cited literature. One example is when we mix ratings from collaborative-filtering approaches, with others structures, such as demographic-based recommendations. In this situation, we have a system that collect users feedback in rating form, and another that starts from an assumption that people with similar demographic attributes share preferences. An approach that consider demographic attributes and rating similarities, although very powerful, can be very trick to deal with.

Another example of challenge, is presented in Zhu et al. (2021). That paper explores popularity bias on dynamic recommendation. Basically it is exposed the problem of popular items being excessively exhibited to users, instead of less popular items, that the user may like.

Regarding the limitations, we have explored in this paper some of them, such as the sparsity problem that often happens on CF systems, due the size of datasets. We also presented the scalability problem, that occurs with a extensive grow of users, etc.

But there is an interesting limitation of recommendation systems, the cold start problem. This problem takes place in situations where we have a new user or a new item. Due the fact that we have not much previous data about this user or item, it is faced a problem, where the system cannot provide useful recommendations.

References

Adomavicius, G., Sankaranarayanan, R., Sen, S., & Tuzhilin, A. (2005, jan). Incorporating contextual information in recommender systems using a multidimensional approach. *ACM Trans. Inf. Syst.*, *23*(1), 103–145. Retrieved from https://doi.org/10.1145/1055709.1055714 doi: 10.1145/1055709.1055714

Adomavicius, G., & Tuzhilin, A. (2005). 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*, *17*(6), 734–749.

Aggarwal, C. C., Wolf, J. L., Wu, K.-L., & Yu, P. S. (1999). Horting hatches an egg: A new graph-theoretic approach to collaborative filtering. In *Proceedings of the fifth acm sigkdd international conference on knowledge discovery and data mining* (pp. 201–212).

Agrawal, R., Imieliński, T., & Swami, A. (1993). Mining association rules between sets of items in large databases. In *Proceedings of the 1993 acm sigmod international conference on management of data* (pp. 207–216).

Agrawal, R., Srikant, R., et al. (1994). Fast algorithms for mining association rules. In *Proc. 20th int. conf. very large data bases, vldb* (Vol. 1215, pp. 487–499).

Amazon machine learning. (n.d.). https://aws.amazon.com/pt/machine-learning/. (Accessed: 2022-10-21)

Balabanović, M. (1997). An adaptive web page recommendation service. In *Proceedings of the first international conference on autonomous agents* (pp. 378–385).

Balabanović, M., & Shoham, Y. (1997). Fab: content-based, collaborative recommendation. *Communications of the ACM*, *40*(3), 66–72.

Barkan, O., & Koenigstein, N. (2016). Item2vec: neural item embedding for collaborative filtering. In *2016 ieee 26th international workshop on machine learning for signal processing (mlsp)* (pp. 1–6).

Bell, R. M., & Koren, Y. (2007). Scalable collaborative filtering with jointly derived neighborhood interpolation weights. *Seventh IEEE International Conference on Data Mining (ICDM 2007)*, 43-52.

Benouaret, I., & Lenne, D. (2016). A package recommendation framework for trip planning activities. In *Proceedings of the 10th acm conference on recommender systems* (p. 203–206). New York, NY, USA: Association for Computing Machinery. Retrieved from https://doi.org/10.1145/2959100.2959183 doi: 10.1145/2959100.2959183

Billsus, D., Pazzani, M. J., et al. (1998). Learning collaborative information filters. In *Icml* (Vol. 98, pp. 46–54).

Blum, A. (1997). Empirical support for winnow and weighted-majority algorithms: Results on a calendar scheduling domain. *Machine Learning*, *26*(1), 5–23. doi: 10.1023/A:1007335615132

Brand, M. (2003). Fast online svd revisions for lightweight recommender systems. In *Proceedings of the 2003 siam international conference on data mining* (pp. 37–46).

Brown, P., Bovey, J., & Chen, X. (1997). Context-aware applications: from the laboratory to the marketplace. *IEEE Personal Communications*, *4*(5), 58-64. doi: 10.1109/98.626984

Canny, J. (2002). Collaborative filtering with privacy via factor analysis. In *Proceedings of the 25th annual international acm sigir conference on research and development in information retrieval* (pp. 238–245).

Cena, F., Console, L., & Vernero, F. (2021). Logical foundations of knowledge-based recommender systems: A unifying spectrum of alternatives. *Information Sciences*, *546*, 60-73. Retrieved from https://www.sciencedirect.com/science/article/pii/S0020025520307568 doi: https://doi.org/10.1016/j.ins.2020.07.075

Channarong, C., Paosirikul, C., Maneeroj, S., & Takasu, A. (2022). Hybridbert4rec: A hybrid (content-based filtering and collaborative filtering) recommender system based on bert. *IEEE Access*, *10*, 56193-56206.

DeCoste, D. (2006). Collaborative prediction using ensembles of maximum margin matrix factorizations. In *Proceedings of the 23rd international conference on machine learning* (pp. 249–256).

Deshpande, M., & Karypis, G. (2004). Item-based top-n recommendation algorithms. *ACM Transactions on Information Systems (TOIS)*, *22*(1), 143–177.

Felfernig, A., & Burke, R. (2008). Constraint-based recommender systems: Technologies and research issues. In *Proceedings of the 10th international conference on electronic commerce.* New York, NY, USA: Association for Computing Machinery. Retrieved from https://doi.org/10.1145/1409540.1409544 doi: 10.1145/1409540.1409544

Getoor, L., Friedman, N., Koller, D., Pfeffer, A., & Taskar, B. (2007). Probabilistic relational models. *Introduction to statistical relational learning*, *8*.

Getoor, L., Sahami, M., et al. (1999). Using probabilistic relational models for collaborative filtering. In *Workshop on web usage analysis and user profiling (webkdd'99)* (pp. 1–6).

Goldberg, D., Nichols, D., Oki, B. M., & Terry, D. (1992). Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, *35*(12), 61–70.

Greening, D. (1997). Building consumer trust with accurate product recommendations. *LikeMinds White Paper LMWSWP-210-6966*.

Guo, G., Zhang, J., & Yorke-Smith, N. (2013). A novel bayesian similarity measure for recommender systems. In *Proceedings of the 23rd international joint conference on artificial intelligence (ijcai)* (p. 2619-2625).

Gupta, D., Digiovanni, M., Narita, H., & Goldberg, K. (1999). Jester 2.0 (poster abstract) evaluation of an new linear time collaborative filtering algorithm. In *Proceedings of the 22nd annual international acm sigir conference on research and development in information retrieval* (pp. 291–292).

Hashemi, S. M., & Rahmati, M. (2020, dec). Cross-domain recommender system using generalized canonical correlation analysis. *Knowledge and Information Systems*, *62*(12), 4625–4651. Retrieved from https://link.springer.com/10.1007/s10115-020-01499-4 doi: 10.1007/s10115-020-01499-4

He, X., Liao, L., Zhang, H., Nie, L., Hu, X., & Chua, T.-S. (2017). Neural collaborative filtering. *Proceedings of the 26th International Conference on World Wide Web*.

Hofmann, T. (2004). Latent semantic models for collaborative filtering. *ACM Transactions on Information Systems (TOIS)*, *22*(1), 89–115.

Ibm watson. (n.d.). https://www.ibm.com/lets-create/. (Accessed: 2022-10-21)

Kautz, H., Selman, B., & Shah, M. (1997, mar). Referral web: Combining social networks and collaborative filtering. *Commun. ACM*, *40*(3), 63–65. Retrieved from https://doi.org/10.1145/245108.245123 doi: 10.1145/245108.245123

Kim, D., & Yum, B.-J. (2005). Collaborative filtering based on iterative principal component analysis. *Expert Systems with Applications*, *28*(4), 823–830.

Koren, Y. (2008). Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th acm sigkdd international conference on knowledge discovery and data mining* (pp. 426–434).

Koren, Y., Bell, R., & Volinsky, C. (2009). Matrix factorization techniques for recommender systems. *Computer*, *42*(8), 30–37.

Koren, Y., & Sill, J. (2013). Collaborative filtering on ordinal user feedback. In *Twenty-third international joint conference on artificial intelligence* (p. 3022-3026).

Krulwich, B. (1996). Learning user information interests through extraction of semantically significant phrases. In *Proceedings of the aaai spring symposium on machine learning in information access, 1996.*

Lang, K. (1995). Newsweeder: Learning to filter netnews. In *Machine learning proceedings 1995* (pp. 331–339). Elsevier.

Lemire, D., & Maclachlan, A. (2005). Slope one predictors for online rating-based collaborative filtering. In *Sdm.*

Lin, W., Alvarez, S. A., & Ruiz, C. (2002). Efficient adaptive-support association rule mining for recommender systems. *Data mining and knowledge discovery*, *6*(1), 83–105.

Linden, G., Smith, B., & York, J. (2003). Amazon. com recommendations: Item-to-item collaborative filtering. *IEEE Internet computing*, *7*(1), 76–80.

Little, R. J., & Rubin, D. B. (2019). *Statistical analysis with missing data.* John Wiley & Sons.

Littlestone, N., & Warmuth, M. K. (1994). The Weight Majority Algorithm. *Information and Computation*(108), 212–261.

Liu, B., Hsu, W., & Ma, Y. (1999). Mining association rules with multiple minimum supports. In *Proceedings of the fifth acm sigkdd international conference on knowledge discovery and data mining* (pp. 337–341).

Masthoff, J. (2004, 01). Selecting news to suit a group of criteria: An exploration.

Melville, P., & Sindhwani, V. (2010). Recommender systems. *Encyclopedia of machine learning*, *1*, 829–838.

Mild, A., & Natter, M. (2002). Collaborative filtering or regression models for internet recommendation systems? *Journal of Targeting, Measurement and Analysis for marketing*, *10*(4), 304–313.

Miyahara, K., & Pazzani, M. J. (2000). Collaborative filtering with the simple bayesian classifier. In *Pacific rim international conference on artificial intelligence* (pp. 679–689).

O'Connor, M., & Herlocker, J. (1999). Clustering items for collaborative filtering. In *Proceedings of the acm sigir workshop on recommender systems* (Vol. 128).

Park, Y., Park, S., Jung, W., & Lee, S. G. (2015). Reversed CF: A fast collaborative filtering algorithm using a k-nearest neighbor graph. *Expert Systems with Applications*, *42*(8), 4022–4028. Retrieved from http://dx.doi.org/10.1016/j.eswa.2015.01.001 doi: 10.1016/j.eswa.2015.01.001

Pazzani, M., & Billsus, D. (1997). Learning and revising user profiles: The identification of interesting web sites. *Machine learning*, *27*(3), 313–331.

Pazzani, M. J. (1999). A framework for collaborative, content-based and demographic filtering. *Artificial intelligence review*, *13*(5), 393–408.

Pazzani, M. J., & Billsus, D. (2007). Content-based recommendation systems. In *The adaptive web* (pp. 325–341). Springer.

Pazzani, M. J., Muramatsu, J., Billsus, D., et al. (1996). Syskill & webert: Identifying interesting web sites. In *Aaai/iaai, vol. 1* (pp. 54–61).

Quinlan, J. R. (1986). Induction of decision trees. *Machine learning*, *1*(1), 81–106.

Rennie, J. D., & Srebro, N. (2005). Fast maximum margin matrix factorization for collaborative prediction. In *Proceedings of the 22nd international conference on machine learning* (pp. 713–719).

Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., & Riedl, J. (1994). Grouplens: An open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 acm conference on computer supported cooperative work* (pp. 175–186).

Resnick, P., & Varian, H. R. (1997). Recommender systems. *Communications of the ACM*, *40*(3), 56–58.

Rich, E. (1979). User modeling via stereotypes. *Cognitive science*, *3*(4), 329–354.

Salakhutdinov, R., Mnih, A., & Hinton, G. (2007). Restricted boltzmann machines for collaborative filtering. In *Proceedings of the 24th international conference on machine learning* (pp. 791–798).

Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. (2000). *Application of dimensionality reduction in recommender system-a case study* (Tech. Rep.). Minnesota Univ Minneapolis Dept of Computer Science.

Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. (2001). Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on world wide web* (pp. 285–295).

Sarwar, B. M., Konstan, J. A., Borchers, A., Herlocker, J., Miller, B., & Riedl, J. (1998). Using filtering agents to improve prediction quality in the grouplens research collaborative filtering system. In *Proceedings of the 1998 acm conference on computer supported cooperative work* (pp. 345–354).

Schafer, J. B., Konstan, J., & Riedi, J. (1999). Recommender systems in e-commerce. In *Proceedings of the 1st acm conference on electronic commerce - ec '99* (pp. 158–166). New York, New York, USA: ACM Press. Retrieved from http://portal.acm.org/citation.cfm?doid=336992.337035 doi: 10.1145/336992.337035

Schilit, B., & Theimer, M. (1994). Disseminating active map information to mobile hosts. *IEEE Network*, *8*(5), 22-32. doi: 10.1109/65.313011

Shardanand, U., & Maes, P. (1995). Social information filtering: Algorithms for automating "word of mouth". In *Proceedings of the sigchi conference on human factors in computing systems* (pp. 210–217).

Srebro, N., Rennie, J., & Jaakkola, T. (2004). Maximum-margin matrix factorization. *Advances in neural information processing systems*, *17*.

Su, X., & Khoshgoftaar, T. M. (2009). A survey of collaborative filtering techniques. *Advances in artificial intelligence*.

Terveen, L., Hill, W., Amento, B., McDonald, D., & Creter, J. (1997, mar). Phoaks: A system for sharing recommendations. *Commun. ACM*, *40*(3), 59–62. Retrieved from https://doi.org/10.1145/245108.245122 doi: 10.1145/245108.245122

Thingssolver. (n.d.). *Introduction to recommender systems.* Retrieved from https://thingsolver.com/introduction-to-recommender-systems/

Ungar, L. H., & Foster, D. P. (1998). Clustering methods for collaborative filtering. In *Aaai workshop on recommendation systems* (Vol. 1, pp. 114–129).

Vatsal. (2021). *Recommendation Systems Explained.* Retrieved from https://towardsdatascience.com/recommendation-systems-explained-a42fc60591ed

Vucetic, S., & Obradovic, Z. (2005). Collaborative filtering using a regression-based approach. *Knowledge and Information Systems*, *7*(1), 1–22.

Weimer, M., Karatzoglou, A., Le, Q., & Smola, A. (2007). Cofi rank-maximum margin matrix factorization for collaborative ranking. *Advances in neural information processing systems*, *20*.

Xia, Z., Dong, Y., & Xing, G. (2006). Support vector machines for collaborative filtering. In *Proceedings of the 44th annual southeast regional conference* (pp. 169–174).

Yassine, A., Mohamed, L., & Al Achhab, M. (2021, feb). Intelligent recommender system based on unsupervised machine learning and demographic attributes. *Simulation Modelling Practice and Theory*, *107*, 102198. Retrieved from https://linkinghub.elsevier.com/retrieve/pii/S1569190X20301374 doi: 10.1016/j.simpat.2020.102198

Zhang, S., Wang, W., Ford, J., & Makedon, F. (2006). Learning from incomplete ratings using non-negative matrix factorization. In *Proceedings of the 2006 siam international conference on data mining* (pp. 549–553).

Zhang, T., & Iyengar, V. S. (2002). Recommender systems using linear classifiers. *The Journal of Machine Learning Research*, *2*, 313–334.

Zhang, Y., Lai, G., Zhang, M., Zhang, Y., Liu, Y., & Ma, S. (2014). Explicit factor models for explainable recommendation based on phrase-level sentiment analysis. In *Proceedings of the 37th international acm sigir conference on research development in information retrieval* (p. 83–92). New York, NY, USA: Association for Computing Machinery. Retrieved from https://doi.org/10.1145/2600428.2609579 doi: 10.1145/2600428.2609579

Zhu, Z., He, Y., Zhao, X., & Caverlee, J. (2021). Popularity bias in dynamic recommendation. In *Proceedings of the 27th acm sigkdd conference on knowledge discovery & data mining* (pp. 2439–2449).

Ziegler, C.-N. (2006). *Applying feed-forward neural networks to collaborative filtering* (Unpublished doctoral dissertation). Master's Thesis, Universitat Freiburg.

Ziegler, C.-N., McNee, S. M., Konstan, J. A., & Lausen, G. (2005). Improving recommendation lists through topic diversification. In *Proceedings of the 14th international conference on world wide web* (pp. 22–32).