International Journal of Computer Information Systems and Industrial Management Applications.
ISSN 2150-7988 Volume 14 (2022) pp.213-225
© MIR Labs, www.mirlabs.net/ijcisim/index.html

Received: 27 December, 2021; Accepted: 5 April, 2022; Publish: 24 April, 2022

A New Cascade-hybrid Recommender System approach tailored for the Retail Market

Miguel Ângelo Rebelo[1,2], **Duarte Coelho**[1,4], **Fábio Fernandes**[1] **and Ivo Pereira**[1,3,4]

[1]E-goi, Av. Menéres 840,
4450-190 Matosinhos, Portugal
{*mrebelo, dcoelho, ffernandes, ipereira*}@*e-goi.com*

[2]i3s, Rua Alfredo Allen 208,
4200-135 Porto, Portugal
mail@miguelrebelo.com

[3]University Fernando Pessoa, Praça 9 de Abril 349
4249-004 Porto, Portugal
ivopereira@ufp.edu.pt

[4]Interdisciplinary Studies Research Center,
Rua Dr. António Bernardino de Almeida 431,
4200-072 Porto, Portugal

Abstract: **Recommender systems ought to increase profit from product sales, by carefully recommending selected items to users. For this, recommendations need to be relevant, novel and diverse. There are many approaches to this problem, each with its own advantages and shortcomings. This paper proposes a novel way to combine model, memory and content-based approaches in a *cascade-hybrid* system, where each step refines the previous one, sequentially. In addition to this, a straightforward way to easily incorporate time-awareness into rating matrices is also advanced. This hybrid system focuses on being intuitive, flexible, robust, auditable and avoid heavy performance costs. Evaluation metrics such as *Novelty Score* are also formalized and computed, in conjunction with *Catalog Coverage* and *mean recommendation price* to better capture the recommender's performance.**

Keywords: Cascade-hybrid, Recommender System, Intelligent Marketing, Retail, Novelty Score, Catalog Coverage

I. Introduction

Recommender systems (RS) technology has gained momentum as the Web established itself as the medium for electronic transactions, business related or not [2, 19]. Feedback from the customer side is known as ratings, which can be *explicit*, if the customer gives a rating to specify their likes and dislikes, or *implicit*, if one does not have that direct response from the customer but instead have clicks, time-watched or bought products, since the simple act of a user buying or browsing an item may be viewed as an endorsement for that item [16]. RS utilize these various sources of data to infer customer interests, because past proclivities are often good indicators of future choices [2, 19]. Different models approach the problem in slightly different ways.

The first approach is to predict the rating for a user-item pair. For m users and n items, this corresponds to an incomplete $m \times n$ matrix, where the observed rating values are used for training. The missing values are then predicted using this model (also referred to as the matrix completion problem). For a merchant what is valuable here is to identify the *top-k* items for a particular user, or determine the *top-k* users to target for a particular item. In this second case the *top-k* can be derived by solving the first formulation for various user-item combinations and then ranking the predictions [2, 19, 5].

It is important to keep in mind that the primary goal of a RS is to increase product sales (their profit). By recommending carefully selected items to users, RS bring relevant items to the attention of users.

In order to achieve the broader business-centric goals, the technical goals of recommender systems are as follows:

1. Relevance: a RS have to recommend items that are relevant to the user at hand, otherwise, it will be ignored (or worst) [9].
2. Novelty: RS can be win-win if it recommends items that the user has not seen in the past. The opposite, in fact, can lead to reduction in sales [9].
3. Serendipity: Serendipitous recommendations are truly surprising to the user, rather than simply something they did not know about before. Many users may only be consuming items of a specific type, although they may

often have latent interests for items of other types which the users themselves might find surprising. This tricky to tune characteristic can increase sales diversity or begin a new trend of interest in the user. Serendipity has long-term and strategic benefits to the merchant [14].

4. Diversity: The recommended list should contain items of different types, because that raises the probability that the user might like at least one of the items [2, 19].

From the perspective of the users, recommendations can be extremely helpful and improve overall user satisfaction with the web site. Amazon.com was one of the pioneers in RS, being one of the few retailers that had the foresight to realize the potential of this technology. This contributed to its expansion from being a book retailer to selling virtually all kinds of products [5].

This paper extends an initial published work [27] and is organized as follows: section 2 describes the related work, focusing on RS models and types of ratings; section 3 describes the methods, detailing the proposed approach; section 4 presents the obtained results and provides a critic discussion; and section 5 presents some conclusions and future work.

II. Related Work

This section describes the related word focusing on RS models and types of ratings.

A. RS Models

The basic models for RS work with two kinds of data, which are:

1. user-item interactions, such as ratings or buying behaviour. These methods are referred to as collaborative filtering (CF) methods [31].

2. user and/or item variables, such as age and gender or product descriptions and keywords. These are called content/contextual-based RS [24, 23].

Some combine these two approaches to create hybrid systems to perform more robustly in a wide variety of settings [6, 10, 30].

1) CF models

Because most users only view and buy a very small fraction of the available items, most of the ratings are missing, which translates into sparse rating matrices. That poses a challenge to the design of systems that leverage the community [31, 26]. CF methods branch into two general categories:

- Memory-based methods (neighbourhood-based CF), in which the user-item ratings are predicted on the basis of their neighbours. They can be user-based, when the similarity functions are computed between users, or item-based, if the similarity functions are computed between items. They however have trouble with sparse matrices [1].
- Model-based methods, develop a model from user ratings. There are two main approaches to develop these models, which are probability approach or rating prediction. To achieve this, ML techniques such as classification, clustering, and rule-based approaches are used. Model-based approaches tend to have better predictions than memory-based, plus it is capable of handling the problem of sparsity and scalability better. However, model-based approaches requires more time and memory [1, 32].

Recently, it has been shown that some combinations of memory-based and model-based methods provide very accurate results [20].

2) Content-based models

In content-based RS, the content information about the items previously rated by a user is analysed to build a model/profile of user interests [22, 23]. They have some advantages in making recommendations for new items with seldom to none interactions, because other items with similar attributes might have been rated by the user at hand [2, 19]. But due to the fact that the community knowledge is not leveraged, these methods provide *obvious* recommendations, which tends to reduce the diversity of the recommended items, which is an undesirable outcome [14]. This problem is referred to as overspecialization. It is always desirable to have a certain amount of novelty and serendipity in the recommendations. And although they are effective at providing recommendations for new items, they do not work for new users [22, 23, 2, 19].

These methods have different trade-offs from CF, and are therefore useful in certain cold-start scenarios. Despite the disadvantages associated with content-based systems, they often complement collaborative systems quite well because of their ability to leverage content-based knowledge. This complementary behaviour is often leveraged in hybrid RS [22, 6, 35, 7, 10].

In case-based recommender systems [24], specific cases are specified by the user as targets or anchor points. Similarity metrics are then defined on the item attributes to retrieve similar items to these cases [2, 19, 19]. Similarly, this approach from knowledge-based systems can be applied to content-based ones, where instead of the user defining the targets explicitly, the items are defined by their buying history.

3) Context-based: Post-filtering

Contextual information about interactions can also be leveraged to fine-tune recommendations. Such contextual information could include time, location, or social data. For example, the types of clothes recommended by a retailer might depend both on the season and the location of the customer [2, 19, 19].

Some clients may have an issue recommending certain products. What one can do is extract them *ad hoc*, after the final recommendations have been computed, by masking the unwanted products. This way the full information is presented to train the model but then only the most relevant choices are kept. This is a process present in our approach.

4) Hybrid Systems

As the three aforementioned systems exploit different sources of input that may work well in different scenarios, many opportunities exist for hybridization, where various aspects from different types of systems are combined to achieve the best of all worlds [35, 19, 7, 10]. Our approach fits into a group of recommenders called *cascade hybrids*, in which each recommender actively refines the recommendations made by the previous recommender [6, 11] The first recommender can act as a powerful filter and provide a rough ranking, eliminating many of the potential items. The second level of recommendation then uses this rough ranking to further refine it. The resulting ranking is then presented to the user. An example of such a recommender system is EntreeC [6]. The combination of CF and content-based approaches in a way that resolves the drawbacks of each other has demonstrated to improve recommendations when compared to each individual approach [10].

B. Types of Ratings

Rating systems heavily influence the performance of RS. Ratings are specified on a scale that indicates the "level of like" of an item. Rating matrices can also be called utility matrices, if their data refers to the amount of profit or other quantities [34]. They can be continuous (such as in the the Jester joke recommendation engine), interval-based (where a discrete set of ordered numbers - the number of stars in IMDB - are used to quantify like or dislike) or binary (where the user only represents like or dislike, i.e. 1 or 0) [13]. Then, they can be further clustered into two main kinds:

- *explicit feedback*, where the user as to explicitly rate in a predefined scale their level of satisfaction. In *explicit* feedback matrices, ratings correspond to highly discriminant preferences, which makes it easier to apply RS [2, 19].
- *implicit feedback*, as is the case of unary data, where the customer preferences are derived from user interactions with the items, such as the buying behaviour (bought or not) or watch time (continuous) [16, 34]. However, the act of not buying an item does not always indicate a dislike. Due to the lack of information available about whether a user dislikes an item and the fact that there is often not a sufficient level of discrimination between the various observed values of the ratings, RS using this type of matrices have to be dealt with care [2, 19].

Because only a small fraction of the items are rated frequently (referred to as popular items) the distribution of ratings among items often satisfies the *long-tail* property (Fig. 1). This translates into a highly skewed distribution since most of the items are rated only a small number of times [4, 36, 25]. This has important implications:

1. Most popular items tend to be relatively competitive and leave little profit for the merchant. It is argued that many companies, like Amazon, make most of their profit in the long-tail [4];
2. Many RS tend to suggest popular items, due to the difficulty of providing robust rating predictions for less frequent items [8];
3. Because of the differences in the rating patterns of the two types of items, high-frequency ones are not representative of the low-frequency [2, 19].

More meaningful predictions should be obtained by adjusting the RS to take real-world properties, such as sparsity and the long-tail, into account [8, 36, 25].

III. Methods

A. Neighbourhood model-based approach: ALS

Latent factor models are considered to be state-of-the-art in RS [2, 19]. They use dimensionality reduction methods in order to estimate the matrix, as a direct method for matrix completion, by exploiting the fact that significant portions of the rows and columns of data matrices are highly correlated. As a result, the rating matrix can be approximated quite well by a low-rank matrix. So, the goal is to find a set of latent vectors, in which the average squared distance of the data points from the hyper-plane defined by these latent vectors is as small as possible. It is important to keep in mind that latent factor models will simply not work if the data does not have any correlations. One general way of capturing such correlations between columns (or rows) by approximating a matrix is through matrix factorization [3, 29]. This way, a matrix *R* with rank larger than *k*, can often be approximately expressed as the product of rank-k factors:

$$R \approx UV^T \ (1)$$

The error of the approximation is equal to $\|R - UV^T\|^2$, where $\|\cdot\|^2$ represents the sum of the squares of the entries in the resulting residual matrix. Once the matrices U and V have been estimated, the entire ratings matrix can be estimated, which provides all the missing ratings. The importance of the regularization term in the objective function is to reduce the tendency of the model to over-fit at the expense of introducing a bias in the model (thus reducing variance) [37, 18, 16].

There are some very efficient algorithms for the optimization of the objective function. The alternating least squares approach (ALS) is generally more stable than methods such as stochastic gradient descent, thus being less sensitive to both the initialization and the way in which the step sizes are chosen. Because the least-squares problem for each item is independent, this step can be parallelized easily [18, 37].

The key differences among various matrix factorization methods arise in terms of the constraints imposed on U and V (e.g., orthogonality or non–negativity of the latent vectors) and the nature of the objective function (e.g., minimizing the Frobenius norm) [2, 19].

This linear algebra technique can help us discover latent features underlying the interactions between users and items. These latent features give a more compact representation of user tastes and item descriptions. Matrix factorization is particularly useful for very sparse data and can enhance the quality of recommendations [21]. The algorithm works by factorizing the original user-item matrix into two factor matrices:

- user-factor matrix (n_{users}, k)
- item-factor matrix (k, n_{items})

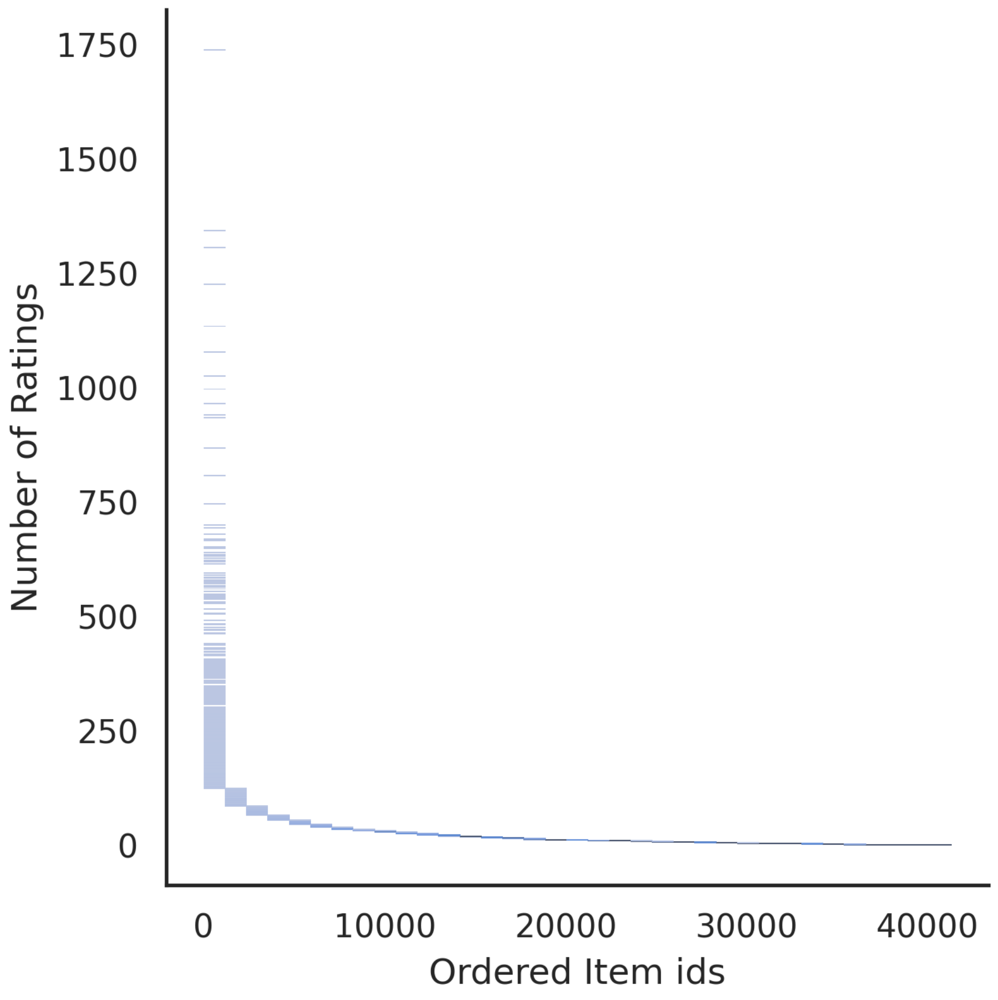

Figure. 1: Long-tail property view on the data set at hand.

It reduces the dimensions of the original matrix into "taste" dimensions. It is very unlikely that one can interpret what each latent feature k represents. However, it is possible to imagine that one latent feature may represent users who like sweatshirts for girls, while another latent feature may represent items which are related to ages between 6 and 12 [2, 19].

In the case of ALS, only one feature vector needs to solve at a time, which means it can be run in parallel (this is a great computational advantage). To do this, U is randomly initialized to solve for V. Then go back and solve for U using the solution for V, then keep iterating back and forth like this until convergence that approximates R as best as possible [18, 37, 16].

In other words, preferences are assumed to be the inner products: $p_{ui} = x_u^T y_i$. These vectors will be known as the user-factors and the item-factors, respectively. Essentially, the vectors strive to map users and items into a common latent factor space where they can be directly compared. This is similar to matrix factorization techniques which are popular for *explicit* feedback data, with two important distinctions:

1. Account for the varying confidence levels (c).

2. Optimization should account for all possible u, i pairs, rather than only those corresponding to observed data.

Accordingly, factors are computed by minimizing the following cost function:

$$\min_{x\star,y\star} \sum_{u,i} c_{ui}(p_{ui} - x_u^T y_i)^2 + \lambda \left(\sum_u ||x_u||^2 + \sum_i ||y_i||^2 \right) \text{ (2)}$$

The r_{ui} variables measure the notion of confidence. A set of binary variables p_{ui}, which indicates the preference of user u to item i. The p_{ui} values are derived by binarizing the r_{ui} values.

In other words, if a user u consumed item i ($r_{ui} > 0$), then it indicates that u likes i ($p_{ui} = 1$). On the other hand, if u never consumed i, no preference is assumed ($p_{ui} = 0$). However, our beliefs are associated with greatly varying confidence levels. First, by the nature of the data, zero values of p_{ui} are associated with low confidence, as not taking any positive action on an item can stem from many other reasons beyond not liking it. Or a consumer may buy an item as a gift for someone else, despite not liking the item for himself. Thus, there would be different confidence levels also among items that are indicated to be preferred by the user. In general, the larger r_{ui} grows, stronger the indication that the user indeed likes the item [16]. Consequently, the set of variables, c_{ui}, intend to measure the confidence in observing p_{ui}:

$$c_{ui} = 1 + \alpha r_{ui} \text{ (3)}$$

The rate of increase is controlled by the constant α. In Yifan Hu, et al. experiments [16], setting $\alpha = 40$ was found to produce good results. The best α parameter for this specific case was estimated through a cross-validation routine.

After this has been finished, the dot product of U and V gives the predicted rating would be for a specific user-item interaction, even if there was no prior interaction. This basic methodology was adopted for *implicit* feedback problems by Yifan Hu, et al., which proposed an efficient (weighted) ALS method for the factorization process in order to avoid the computational challenge of handing the large number of zero entries [16].

B. Data partitioning

This paper focuses on a composite solution to the problem, since it would join the better aspects from different approaches. High and Low interaction customers were divided based on the number of buys by user. For higher interaction customers, a technique based on matrix factorization was chosen. Meanwhile for the lower interaction users, item-item K nearest neighbours (K-NN) model fitted best due to the lack of data from the user. The ALS strategy produced acceptable results for users with more than 3 interactions.

- *high interaction data set* had a total of 1066596 ratings, 147884 unique users and 33555 unique items.

- *low interaction data set* had a total of 14267 ratings, 10328 unique users and 7730 unique items.

C. Content-based approach: Cosine Similarity

Content-based systems largely operate in the text domain, working with a wide variety of item descriptions and knowledge about users [23]. The first step one must take is to convert these different types of unstructured data into standardized descriptions [2, 19]. The procedure of the present paper is generically common to how content-based approaches operate, with the exception that here the pre-computed recommendations are then filtered:

1. Pre-processing and feature extraction: descriptions were transformed into a keyword-based vector-space representation. To simplify the word matrix, everything was set to lower case, and further removed numbers, special characters and accents.

2. Content-based learning of user profiles: user-specific model is constructed to predict user interests in items, based on their past buying history. Pairwise cosine similarity was computed between items and removed self similarity (diagonal).

3. Filtering and recommendation: In this step, the learned model from the previous step is used to filter previous recommendations on items for specific users.

The kind of similarity metric chosen is very important in content-based systems in order to retrieve relevant results. In the current approach, the cosine distance metric was the one used for item descriptions. Mathematically, it measures the cosine of the angle between two vectors projected in a multi-dimensional space [15].

$$Similarity = cos(\theta) = \frac{A \cdot B}{||A|| ||B||} \text{ (4)}$$

The resulting similarity ranges from -1 meaning exactly opposite, to 1 meaning exactly the same, with 0 indicating orthogonality or decorrelation, while in-between values indicate intermediate similarity or dissimilarity.

The cosine similarity is advantageous because even if the two similar products are far apart by the Euclidean distance (due to the size of the description) they could still have a smaller angle between them if they contain 'girl'. The smaller the angle, higher the similarity [15, 2, 19]. The way to get the most similar items is detailed in Algorithm 1.

D. Neighbourhood memory-based Approach: Item-KNN

User-based approaches are often harder to scale because of the dynamic nature of users, whereas items usually don't change much, and item-based approaches often can be computed offline and served without constantly re-training [2, 19]. In addiction, the user-based approaches don't perform well for low interaction cases. This is why the item-KNN approach was chosen for the low interaction users. In the item-based approach, in order to make recommendations for target item B, the user's own ratings on neighbouring (i.e., closely related) items are used [28].

K-NN is a perfect go-to model because it does not make any assumptions on the underlying data distribution but it relies on item feature similarity. When K-NN makes inference about an item, it calculates the distance between the target item and every other item in its database, then it ranks its distances and returns the *top-k* nearest neighbours as the most similar item recommendations [28]. In item-based models, peer groups are constructed in terms of items, therefore, similarities need to be computed between items. The basic idea is to leverage the user's own ratings on similar items in the final step of making the prediction.

1) Adjusted Cosine Similarity

Computing similarity using basic cosine measure in item-based case has one drawback - the differences in rating scale between different users are not taken into account. The adjusted cosine similarity subtracts the corresponding user average from each co-rated pair [28]. This is given by:

$$s(i,j) = \frac{\sum_{u \in U}(R_{u,i} - \overline{R_u})(R_{u,j} - \overline{R_u})}{\sqrt{\sum_{u \in U}(R_{u,i} - \overline{R_u})^2}\sqrt{\sum_{u \in U}(R_{u,j} - \overline{R_u})^2}} \quad (5)$$

Although the Pearson correlation can also be used on the columns in the case of the item-based method, the adjusted cosine generally provides superior results.

2) Weighted Average

This method computes the prediction on an item i for a user u by computing the sum of the ratings given by the user on the items similar to i. Each rating is weighted by the corresponding similarity $s_{i,j}$ between items i and j.

$$P_{u,i} = \frac{\sum_{allsimilaritems,N}(S_{i,N} * R_{u,N})}{\sum_{allsimilaritems,N}(|S_{i,N}|)} \quad (6)$$

This method (weighted sum) tries to capture how the user rates similar items. (scaled by the sum of the similarity terms to make sure the prediction is within the predefined range). The greater prediction accuracy of the item-based method is its main advantage [28].

These methods often provide more relevant recommendations because they are using the user's own ratings to perform the recommendation, since similar items are identified to a target item, and the user's own ratings on those items are used to extrapolate the ratings of the target. For example, similar items to a pair of jeans might be a set of other pairs of jeans [2, 19].

E. Time-Sensitive Recommendations: Time-decay approach

In many settings, the user recommendations for an item should evolve with time, as community attitudes and interests change. The rating of an item might be dependent on the specific season, for example, winter clothing should not be recommended during the summer [2, 19].

Any history data has, as the name suggests, the time at which something occurred (like a buy). One can take advantage of this information, and more, when computing an *implicit* rating. To implement a time-decay algorithm, we've used the time-span (in days) since the item was bought:

$$rating_{ui} = \frac{\beta buy + \alpha n_{ui} + \delta p_{ui}}{(elapsedtime + 1)^{gravity}} \quad (7)$$

being n_{ui} the number of items bought and p_{ui} the price of the product i. β, α, δ and $gravity$ stand for the weights attributed to the act of buying, th number of products bought, the price of the product and the time decay element, respectively. The time decay element, using the term $gravity$, indicates how fast an item's rating decays. People's tastes change, so what was once the best thing for a user might not be its favourite now. It is then a good idea to let old events count less than new ones. By changing the parameters, it is feasible to achieve different rating distributions. These have to be adapted to the particular retailer at hand. Various parameter combinations were tested to see what better captured the differences in types of buys. This algorithm was inspired by Hacker News (https://news.ycombinator.com), which uses a somewhat similar approach, putting importance into recent events and not so much on older ones. Appropriately, these types of algorithms are called $time-decayalgorithms$.

This approach has several advantages. The most relevant of each is that by including information about the price of the product, as well as the number of items bought, the preference of the item i by user u con be inferred more accurately. If an user bought 5 units of item i, there is more confidence to assume an higher preference than if the consumer had only bought one. But to much weight in the number of items and less expensive items will probably be favoured (like socks) that are bought more regularly, which is undesirable. To circumvent this issue, the price of the item has to be weighted. The more expensive the item i, the more probable it is that the user u did some research before the decision to buy that item. This means one can more confidently affirm that the user u has a stronger preference for item i. Final computed ratings were re-scaled to fit between 0 and 1 scale, for easier computational strain.

Algorithm 1 get most similar items

```
1:  procedure ALGORITHM
2:  user history ← buying history for each user id
3:  For each user id:
4:      history items ← user history($uid$)
5:      similarity vectors ← similarity matrix[history items($indeces$)]
6:      centroid ← $\overline{similarityvectors}$
7:      recommended items ← user recommends($uid$)
8:      subcentroid ← centroid[recommended items($indeces$)]
9:      $orderthesubcentroid$
10:     product indeces ← top-k indeces from subcentroid (most similar to the history)
11:     product ids ← product ids[product indeces
12:     return product ids
13:     goto loop
14:     close;
```

F. Log-transformed prices

There are some extremely pricey products that can skew the recommendations, when opting for a time-decay algorithm as described before. To control this effect of the price range in the assumptions about the preferences, the prices were log-transformed. This is a classic approach for dealing with skewed data, to decrease the variability of data and make it conform more closely to the normal distribution. While this is not always the case, log-transformations should be done with caution, it is very useful in a situation where we do not have to evaluate causation. There were some products with price equal to 0. To correct this, the median price for that product was calculated and replaced in the missing value. Furthermore, when the logarithm of a price resulted in a negative number, that was replaced with 0.1.

G. Evaluation of RS

The evaluation of RS is akin to that of classification techniques. But its evaluation is often trickier since a single criterion cannot capture many of the goals of the designer [5].

Hypothesis testing is the ideal way to test the performance of a RS. By measuring the conversion rate of users clicking on articles that were recommended, the direct impact of the RS can be estimated. However, since online evaluations require active user participation, it is often not feasible to use them in bench-marking and research. Offline evaluations with historical data sets are then used [2, 19, 12].

The issue is that accuracy measures alone can often provide an incomplete picture of the true conversion rate of a RS. Metrics such as *novelty*, *coverage*, and *serendipity* are important for the user experience and have important short and long-term impacts on the conversion rates. Nevertheless, the actual quantification of some of these factors is often quite subjective, and there are often no hard measures to provide a numerical metric [2, 19, 14, 12].

The measures to evaluate the proposed model were *root mean squared error* RMSE (during hyper-parameter optimization only), *Catalogue Coverage* (CC), *Novelty*, *Recall*, *Popularity* and *Scalability* measures. The accuracy measure chosen for hyper-parameter optimization (RMSE), represents the quadratic mean of the differences between predicted values and observed values of ratings.

$$RMSE(\hat{\theta}) = \sqrt{E((\hat{\theta} - \theta)^2)} \quad (8)$$

The CC, which is specifically suited for recommendation lists, objectively evaluates how the item catalogue is being covered by the recommendations, since they should be diverse across users [12]. Let T_u represent the list of top-k items recommended to user $u \in 1...m$. The CC is defined as the fraction of items that are recommended to at least one user [12].

$$CC = \frac{|U_{u=1}^{m} T_u|}{n} \quad (9)$$

The *Novelty Score* (NS) can be estimated using offline methods, using the rating's time stamps. Novel systems are better at recommending items that are more likely to be selected by the user in the future. All ratings after time t_0 were removed from the training data, which were then used for scoring purposes. For each item rated after t_0 and correctly recommended, the novelty evaluation score is rewarded. It is assumed that popular items are less likely to be novel, and less credit was given for recommending popular items. To compute *recall* only the proportion of correct predictions across users and averaged them was computed.

$$NS = \frac{|U_{u=1}^{m} C_u| - \frac{|U_{u=1}^{m} P_u|}{2}}{n_r} \quad (10)$$

Here, C_u represents the number of correct recommendations in the list of top-k items recommended to user u, P_u is the number of popular items in C_u and n_r is the total number of recommendations. This is inspired by the works of Mouzhi Ge, et al. [12].

A *Popularity Score* (PS) was also computed, which tells us how many of the right predictions were of popular items. It should not be too low since it would mean that the recommendations were being too random.

RS should be designed to perform effectively and efficiently in the presence of large amounts of data [33]. For this purpose, some measures to determine the scalability of a system were collected:

1. Training time
2. Prediction time

3. Memory requirements

With the emergence of the "Big-data" paradigm, these measures have become increasingly relevant.

IV. Results and Discussion

A. *Time-decay Ratings*

Our data set from a fashion retailer revealed that most clients buy less than 0.01% of the product range. with 148301 users and 41285 items in total and $sparcity = 99.98\%$. Most popular items are the less expensive ones, which is a common situation (Fig 2a and 2b). It is not ideal to center the recommendations along these products, as they have lower profit margins [4].

As the empirical cumulative distribution (ECDF) of buys by the price of the product shows, lower priced products represent most of the interactions (Fig. 2a and 2b). It is important to notice that the price upper quartile per user is 14.99, with a mean of 15.27 per item bought. Some products occasionally had $price = 0$. This could be due to promotions. These zeros were replaced with the median price of the product.

Furthermore, extremely costly products are a rarity in this particular data set. The mean number of products bought by each user is 4.51 but there are cases in the hundreds (175 is the max). So, when designing the ratings, this variables have to be carefully weighted, since we ought to obtain highly discriminant implicit preferences without favoring too much the expensive items nor the popular ones. After testing different hyper-parameter combinations for the *time-decay* algorithm, three ratings were chosen (Fig. 3, with similar distribution characteristics.

By including information about the unitary price, as well as the number of items bought, the preference of the item i by the user u can be more accurately inferred [34]. If u bought 5 units of item i, there is more confidence to assume a higher preference than if he had only bought one. But to much weight in the number of items and the less expensive ones will probably be favored, which is undesirable. To circumvent this issue, the price of each item has to be weighted in. The more expensive the item i, the more probable it is that the user u did some research before the decision to buy that item. This means one can more confidently affirm that the user u has a stronger preference for item i. Furthermore, since the data comes from a fashion retailer, the decay of the ratings with time is of high importance to promote the newest (on the season) products. The hyper-parameter spaces tested were: $\beta = [0.9, 0.8, 0.9]$, $\alpha = [0.025, 0.02, 0.01]$, $\delta = [0.075, 0.18, 0.09]$, *gravity* $= [0.2, 0.2, 0.3]$.

B. *Model-based Approach*

The hyper-parameters for the matrix factorization model were optimized for each rating using RMSE as the metric of choice, in a cross-validation routine. Fig. 4 shows the results for the optimal hyper-parameter space obtained for the specific data set at hand, through the cross-validation routine. The set of hyper-parameters tested was the following:

- Number of latent factors: [20, 40, 60, 80]

Distance Metric	RMSE	Train (s)	Test (s)
ItemKNN-Cosine	0.0077	4.4777	1.0177
ItemKNN-Pearson	0.0077	3.3041	1.1570
ItemKNN-AdjustedCosine	0.0077	2.9520	1.0732

Table 1: Distance metrics tests for Item-KNN.

- λ: [0.01, 0.001]
- α: [10, 20, 40, 70]
- Max iterations: [10, 25, 40, 50]

C. *Memory-based Approach*

For the low interaction set of customers, the chosen approach was to compute the K-NN across items, because of the lack of information from the user side. This way one can have accurate recommendations, at the sacrifice of serendipity. Table 1 gives a summary of the tests went through to define the similarity measure to use in this case and the number of k-neighbours to use (not shown).

Although the RMSE was the same for all distance metrics (in this particular data set), the adjusted cosine revealed to be faster to train and faster to predict, and functioned as a deciding factor.

D. *Similarity Pruning*

One key aspect of this cascade approach, and its novelty, is the *similarity pruning* technique used to filter the pre-computed recommendations from other methods. This proved to be valuable, not on decreasing the error but in giving contextually more relevant and accurate recommendations.

The pre-computed latent-model recommendations were filtered using the proposed content-based approach, described in Algorithm 1. Pre-computing using the latent-model double/triple the number of products needed in the end, gave enough diversity and serendipity to the recommendations while the *similarity pruning* assured that sub-optimal suggestions were not being presented (i.e., mixing girl and boy recommendations for a client with a history of only buying boy's clothes).

As with the model-based procedure to recommend for the high interaction users, pre-computed Item-KNN recommendations for the low interaction ones were also filtered using the content-based approach, described in Algorithm 1. The strategy was akin to the one adopted in the previous case, recommending double/triple the number of products needed to then filter based on content similarity. This assured, again, that sub-optimal suggestions were avoided, according to the business context.

E. *Context-based Post-filtering*

The retailer in question imposed some restrictions to some product categories (such as socks), based on the profit margin and other factors. Such contingencies may be necessary based on specific business cases. Although the proposed approach to the rating computation prevents the model from

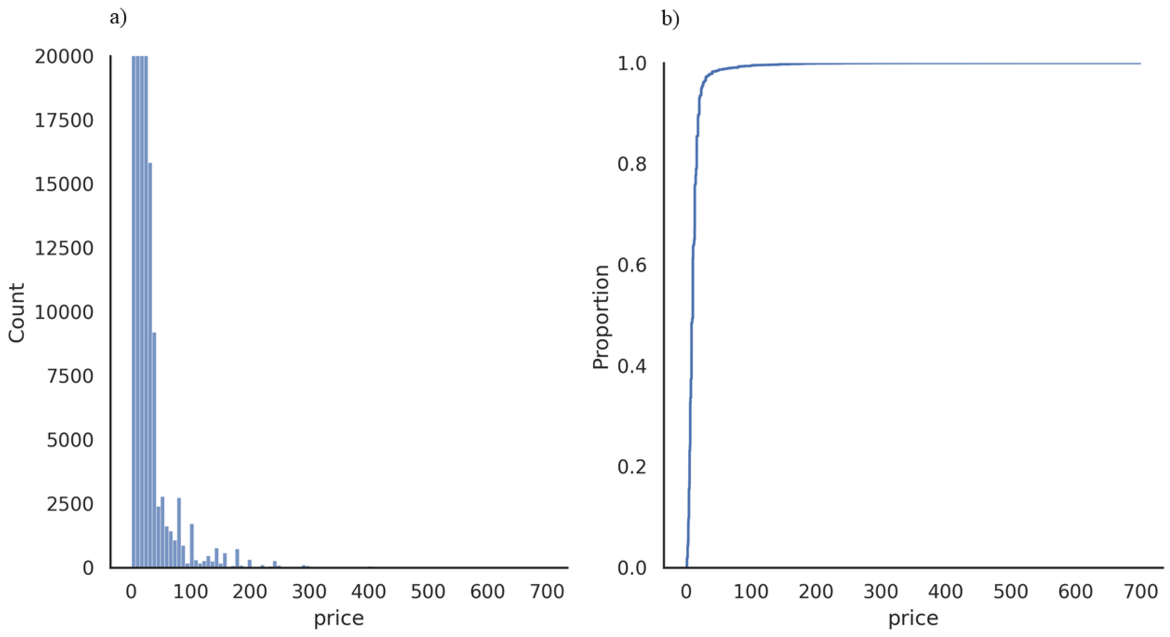

Figure. 2: a) Distribution of buys by price. Lower priced products sell much more. b) ECDF of buys by price. Lower priced products amount to almost all of the buys.

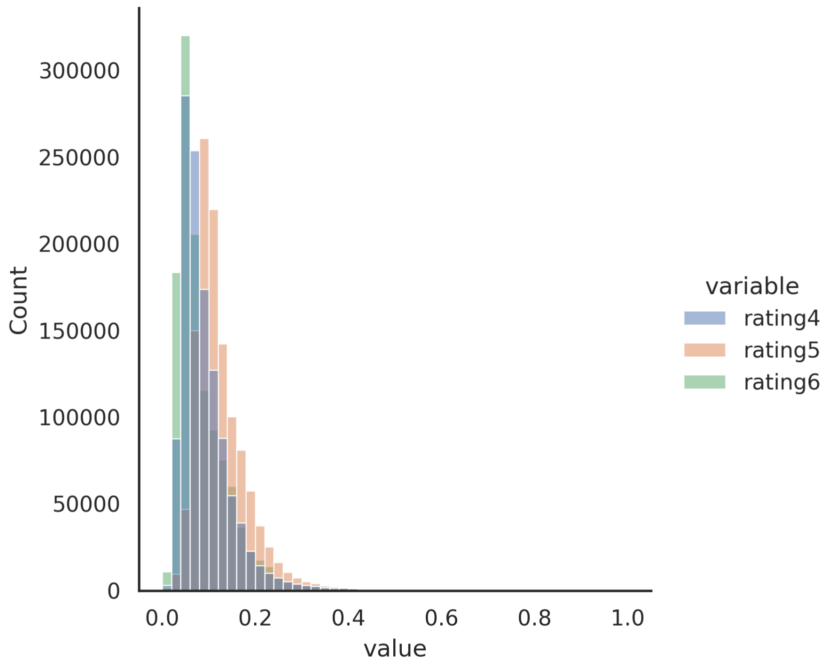

Figure. 3: Computed ratings' distributions.

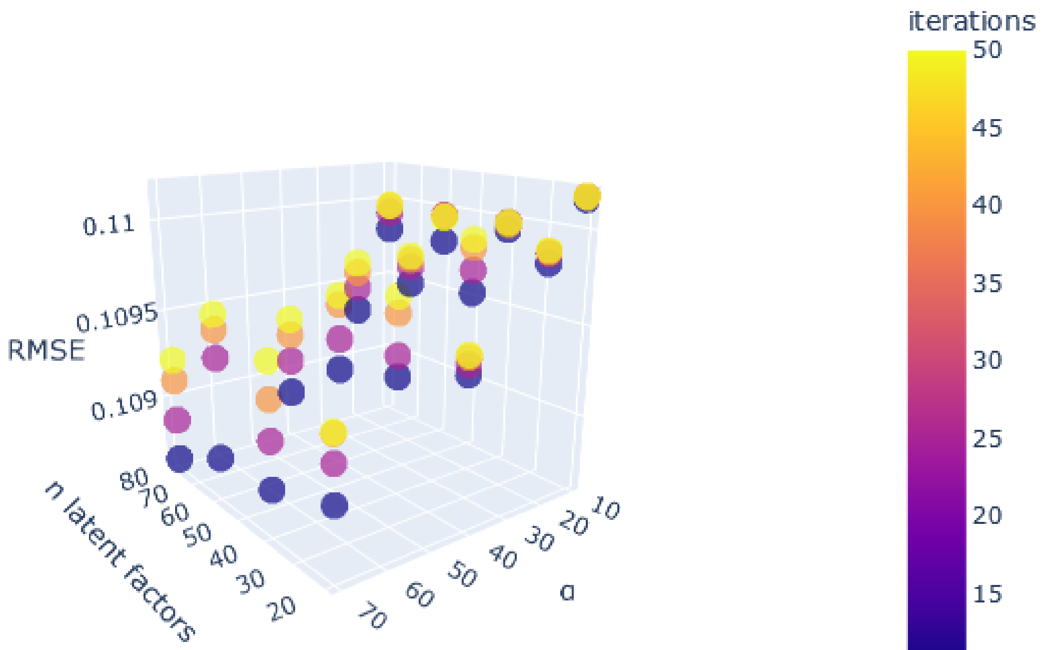

Figure. 4: Optimal hyper-parameter space search.

recommending older items or product with lower profit margins, to assure that they do not appear in the output recommendations, one needs to apply a final filter to the previously ranked list of recommendations and serve the final ones. Such contextual information could also include time, location, or social data. This way the full information was presented to model the data but then only kept the most relevant choices.

F. Evaluation and Performance considerations

By measuring the conversion rate of users, the direct impact of the RS can be estimated. However, it is often not feasible to use them in bench-marking and research. For that reason, offline evaluations with historical data sets are used [2, 19, 12]. Accuracy measures alone can often provide an incomplete picture of the RS's performance. Metrics such as *novelty*, *coverage*, and *serendipity*, though hard to quantify, are important for the user experience and have important short and long-term impacts on the conversion rates [2, 19, 14, 12].

The proposed model was evaluated using *root mean squared error* RMSE (hyper-parameter optimization only) [17], *Catalog Coverage* (CC), *Novelty Score* and *Scalability* measures. The *mean recommendation price* was also computed, since pricier items have higher profit margins. CC evaluates how the item catalog is being covered by the recommendations, since they should be diverse across users [12]. Let T_u represent the list of top-k items recommended to user u. The CC is defined as the fraction of items that are recommended to at least one user [12].

$$CC = \frac{|U_{u=1}^{m} T_u|}{n} \quad (11)$$

The *Novelty Score* (NS) can be estimated using the rating's

Model	Catalog Coverage	Novelty Score	Median Price
baseline	7.51%	6.24%	11.51
Model 1	19.80%	1.58%	11.98
Model 2	19.51%	1.53%	13.36
Model 3	19.43%	1.66%	12.79

Table 2: Model evaluation metrics.

time stamps. All ratings after time t_0 were removed from the training data, and then used for scoring purposes. For each item rated after t_0 and correctly recommended, the novelty evaluation score is rewarded. In this approach it is assumed that popular items are less likely to be novel, and less credit was given to them.

$$NS = \frac{|U_{u=1}^{m} C_u| - \frac{|U_{u=1}^{m} P_u|}{2}}{n_r} \quad (12)$$

Here, C_u represents the number of correct recommendations in the list of items for an user u, P_u is the number of popular items in C_u and n_r is the total number of recommendations, as inspired by the works of Mouzhi Ge, et al. [12].

Table 2 shows the evaluation metrics obtained for the proposed model with three different ratings (computed using different *time-decay* hyper-parameter spaces, detailed above), plus a model using unary ratings (baseline). Due to the nature of this particular data set, the proposed model's NS is lower compared to the baseline. There is a trade-off between CC and NS, so some compromise is necessary. Meanwhile, CC averaged close to 20% across the proposed models. These matrices largely surpass the baseline, which is a good indicator of the ability to recommend across a diverse set of products from the catalog. Fig. 5 demonstrates how CC evolves with the proportion o users that get recommendations. Recommendations using the baseline recommender with the unary ratings hit the *plateau* much quicker, having a distinct disadvantage when compared to the proposed model using the computed ratings. This and the improved *mean recommendation price* have important business implications [9, 14, 2, 19, 4].

RS should also be designed to perform efficiently [33]. For this purpose, some measures of scalability were also collected: Training time was approximately 8 *s*/$200M_{entries}$; Prediction time averaged 30 *s*/1000_{users}; Memory requirements averaged 15 *Gb* for the same matrix.

Both the model-based and memory-based approaches have good performance because they can be easily parallelized. There are very efficient applications using both [28, 16]. Still and all, the present approach is using both and applying on top of it a content-based method. By the data was partitioned between high and low interaction users, the size of the matrices for each approach (model and memory-based) was significantly reduced. So, the same original matrix is not being used twice. One has also to account for the transformation of the product descriptions into vector-space representations, the computation of pairwise similarity between items and then the computational strain from the pruning done by Algorithm 1. This slows down the total computation time, as expected. But due to the simple matrix calculations it performs, it is not slower in total than any of the other methods in isolation. One of its advantages is that it has no hyper-parameters to optimize and works with just the product descriptions. Both the model-based and memory-based methods have hyper-parameters that need to be tailor-optimized for the issue at hand.

V. Conclusions

The present work describes a novel *cascade-hybrid* approach for RS that combines time awareness and strengths from the model, memory and content-based approaches in an intuitive and flexible manner, without heavy performance costs. Offline evaluation metrics for RS were also developed on, formalizing a way to compute NS. The model exhibited good evaluation metrics, revealing a good trade-off between CC and NS. Checking the outputs from the same model using different *time-decay* hyper-parameters, proved that this step has a great influence on final results and needs to be tailor-optimized for the retailer at hand. Though good, there is an ongoing effort to raise NS metrics. This *cascade-hybrid* approach has one key advantage beyond being highly adaptable: it is highly explainable and does not function in a black-box fashion. It can adapt to diverse business practices and objectives, and its results are easily auditable. In the content-based similarity pruning using item's textual descriptions, we are working on giving textual features differential weight based on the importance of product characteristics. There is also a performance cost to this hybrid solution.

Acknowledgments

This article is a result of the project ”Criação de um Núcleo de I&D para a geração de novo conhecimento nas áreas de Inteligência Artificial, Machine Learning, Intelligent Marketing e One-2-One Marketing”, supported by Operational Programme for Competitiveness and Internationalisation (COMPETE 2020), under the PORTUGAL 2020 Partnership Agreement, through the European Regional Development Fund (ERDF), for E-goi.

References

[1] Aditya, P.H., Budi, I., Munajat, Q.: A comparative analysis of memory-based and model-based collaborative filtering on the implementation of recommender system for E-commerce in Indonesia: A case study. In: 2016 International Conference on Advanced Computer Science and Information Systems. pp. 303–308. IEEE, Malang, Indonesia (2016). https://doi.org/10.1109/ICACSIS.2016.7872755

[2] Aggarwal, C.C.: Recommender Systems. Springer International Publishing, Cham (2016). https://doi.org/10.1007/978-3-319-29659-3

[3] Aggarwal, C.C., Parthasarathy, S.: Mining massively incomplete data sets by conceptual reconstruction. In: Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 227–232. KDD '01, Association for Computing Machinery, New York, USA (2001). https://doi.org/10.1145/502512.502543

[4] Anderson, C.: The Long Tail: Why the Future of Business Is Selling Less of More. New York: Hyperion (2006)

[5] Balog, K., Radlinski, F.: Measuring recommendation explanation quality: The conflicting goals of explanations. Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (2020)

[6] Burke, R.: Hybrid recommender systems: Survey and experiments. User Modeling and User-Adapted Interaction **12**(4), 331–370 (2002). https://doi.org/10.1023/A:1021240730564

[7] Çano, E., Morisio, M.: Hybrid recommender systems: A systematic literature review. Intelligent Data Analysis **21**, 1487–1524 (2017)

[8] Cremonesi, P., Koren, Y., Turrin, R.: Performance of recommender algorithms on top-n recommendation tasks. In: Proceedings of the Fourth ACM Conference on Recommender Systems. pp. 39–46. RecSys '10, Association for Computing Machinery, Barcelona, Spain (2010). https://doi.org/10.1145/1864708.1864721

[9] Fleder, D.M., Hosanagar, K.: Recommender systems and their impact on sales diversity. In: Proceedings of the 8th ACM Conference on Electronic Commerce - EC '07. p. 192. ACM Press, San Diego, California, USA (2007). https://doi.org/10.1145/1250910.1250939

[10] Forsati, R., Doustdar, H.M., Shamsfard, M., Keikha, A., Meybodi, M.R.: A fuzzy co-clustering approach for hybrid recommender systems. Int. J. Hybrid Intell. Syst. **10**(2), 71–81 (Apr 2013). https://doi.org/10.3233/HIS-130166

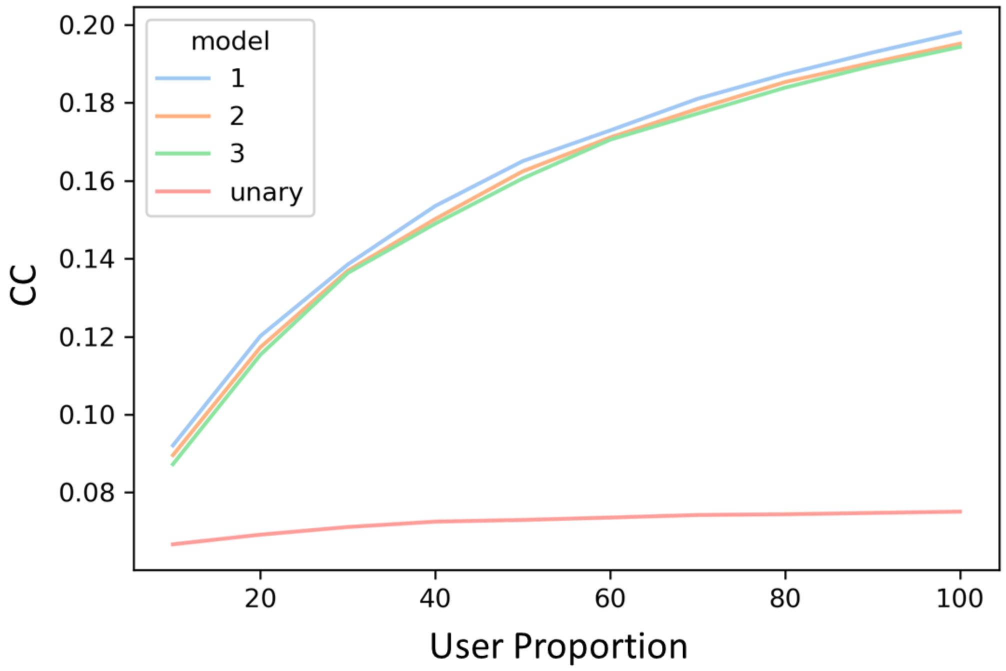

Figure. 5: CC by proportion of users with recommendations.

[11] Funk, P. (ed.): Advances in Case-Based Reasoning: 7th European Conference, ECCBR 2004, Madrid, Spain, August 30 - September 2, 2004; Proceedings. No. 3155 in Lecture Notes in Computer Science Lecture Notes in Artificial Intelligence, Springer, Berlin Heidelberg (2004)

[12] Ge, M., Delgado-Battenfeld, C., Jannach, D.: Beyond accuracy: Evaluating recommender systems by coverage and serendipity. In: The 4th ACM Conference. p. 257. ACM Press, Barcelona, Spain (2010). https://doi.org/10.1145/1864708.1864761

[13] Goldberg, K., Roeder, T., Gupta, D., Perkins, C.: Eigentaste: A Constant Time Collaborative Filtering Algorithm. Information Retrieval **4**(2), 133–151 (Jul 2001). https://doi.org/10.1023/A:1011419012209

[14] Good, N., Schafer, J.B., Konstan, J.A., Borchers, A., Sarwar, B., Herlocker, J., Riedl, J.: Combining collaborative filtering with personal agents for better recommendations. In: Combining Collaborative Filtering with Personal Agents for Better Recommendations. pp. 439–446. AAAI '99/IAAI '99, American Association for Artificial Intelligence, Orlando, Florida, USA (1999)

[15] Han, J., Kamber, M., Pei, J.: Getting to Know Your Data. In: Data Mining, pp. 39–82. Elsevier (2012). https://doi.org/10.1016/B978-0-12-381479-1.00002-2

[16] Hu, Y., Koren, Y., Volinsky, C.: Collaborative Filtering for Implicit Feedback Datasets. In: 2008 Eighth IEEE International Conference on Data Mining. pp. 263–272. IEEE, Pisa, Italy (2008). https://doi.org/10.1109/ICDM.2008.22

[17] Hyndman, R.J., Koehler, A.B.: Another look at measures of forecast accuracy. International Journal of Forecasting **22**(4), 679–688 (2006). https://doi.org/10.1016/j.ijforecast.2006.03.001

[18] Jain, P., Netrapalli, P., Sanghavi, S.: Low-rank matrix completion using alternating minimization. In: Proceedings of the 45th Annual ACM Symposium on Symposium on Theory of Computing - STOC '13. p. 665. ACM Press, Palo Alto, California, USA (2013). https://doi.org/10.1145/2488608.2488693

[19] Ko, H., Lee, S., Park, Y., Choi, A.: A Survey of Recommendation Systems: Recommendation Models, Techniques, and Application Fields. Electronics **11**(1), 141 (Jan 2022). https://doi.org/10.3390/electronics11010141

[20] Koren, Y.: Factor in the neighbors: Scalable and accurate collaborative filtering. ACM Transactions on Knowledge Discovery from Data **4**(1), 1–24 (2010). https://doi.org/10.1145/1644873.1644874

[21] Koren, Y., Bell, R., Volinsky, C.: Matrix Factorization Techniques for Recommender Systems. Computer **42**(8), 30–37 (2009). https://doi.org/10.1109/MC.2009.263

[22] Lops, P., de Gemmis, M., Semeraro, G.: Content-based Recommender Systems: State of the Art and Trends. In: Ricci, F., Rokach, L., Shapira, B., Kantor, P.B. (eds.) Recommender Systems Handbook, pp. 73–105. Springer US, Boston, MA (2011)

[23] Lops, P., Jannach, D., Musto, C., Bogers, T., Koolen, M.: Trends in content-based recommendation. User Modeling and User-Adapted Interaction **29**, 239–249 (2019)

[24] Lorenzi, F., Ricci, F.: Case-Based Recommender Systems: A Unifying View. In: Mobasher, B., Anand, S.S.

(eds.) Intelligent Techniques for Web Personalization, pp. 89–113. Springer Berlin Heidelberg (2005)

[25] Park, Y.J., Tuzhilin, A.: The long tail of recommender systems and how to leverage it. In: Proceedings of the 2008 ACM Conference on Recommender Systems - RecSys '08. p. 11. ACM Press, Lausanne, Switzerland (2008). https://doi.org/10.1145/1454008.1454012

[26] Raj, S.D.: Automated service recommendation with preference awareness: An application of colaborative filtering approach in big data analytics. 2017 2nd International Conference on Communication and Electronics Systems (ICCES) pp. 766–769 (2017)

[27] Rebelo, M.Â., Coelho, D., Pereira, I., Fernandes, F.: A new cascade-hybrid recommender system approach for the retail market. In: Abraham, A., Madureira, A.M., Kaklauskas, A., Gandhi, N., Bajaj, A., Muda, A.K., Kriksciuniene, D., Ferreira, J.C. (eds.) Innovations in Bio-Inspired Computing and Applications. pp. 371–380. Springer International Publishing, Cham (2022)

[28] Sarwar, B., Karypis, G., Konstan, J., Reidl, J.: Item-based collaborative filtering recommendation algorithms. In: Proceedings of the Tenth International Conference on World Wide Web. pp. 285–295. ACM Press, Hong Kong, Hong Kong (2001). https://doi.org/10.1145/371920.372071

[29] Sarwar, B., Karypis, G., Konstan, J., Riedl, J.: Application of Dimensionality Reduction in Recommender System - A Case Study:. Tech. rep., Defense Technical Information Center, Fort Belvoir, VA (2000). https://doi.org/10.21236/ADA439541

[30] Shamshiri, M., Sing, G.O., Kumar, Y.J.: A recommender system based on group method of data handling neural network. International Journal of Computer Information Systems and Industrial Management Applications **11**, 028–038 (2019)

[31] Srifi, M., Oussous, A., Ait Lahcen, A., Mouline, S.: Recommender Systems Based on Collaborative Filtering Using Review Texts—A Survey. Information **11**(6), 317 (2020). https://doi.org/10.3390/info11060317

[32] Su, X., Khoshgoftaar, T.M.: A Survey of Collaborative Filtering Techniques. Advances in Artificial Intelligence pp. 1–19 (2009). https://doi.org/10.1155/2009/421425

[33] Takács, G., Pilászy, I., Németh, B., Tikk, D.: Scalable collaborative filtering approaches for large recommender systems. Journal of Machine Learning Research **10**, 623–656 (2009)

[34] Tas, K., Onder, E., Aktas, M.S.: On the Implicit Feedback Based Data Modeling Approaches for Recommendation Systems. In: 2021 International Conference on Electrical, Communication, and Computer Engineering (ICECCE). pp. 1–6. IEEE, Kuala Lumpur, Malaysia (Jun 2021). https://doi.org/10.1109/ICECCE52056.2021.9514214

[35] Walek, B., Fojtik, V.: A hybrid recommender system for recommending relevant movies using an expert system. Expert Systems With Applications **158**, 113452 (2020)

[36] Yin, H., Cui, B., Li, J., Yao, J., Chen, C.: Challenging the long tail recommendation. Proc. VLDB Endow. **5**(9), 896–907 (May 2012). https://doi.org/10.14778/2311906.2311916

[37] Zhou, Y., Wilkinson, D., Schreiber, R., Pan, R.: Large-Scale Parallel Collaborative Filtering for the Netflix Prize. In: Fleischer, R., Xu, J. (eds.) Algorithmic Aspects in Information and Management, pp. 337–348. Springer Berlin Heidelberg, Berlin, Heidelberg (2008). https://doi.org/10.1007/978-3-540-68880-832

Author Biographies

Miguel Ângelo Rebelo was born in Espinho, Portugal, 1995. He got his First degree in Biochemistry, at the Faculty of Sciences, University of Porto (2016); Master's degree Molecular Biology and Biotechnology, at the School of Sciences, University of Minho (2018); Specialization in Computational Statistics and Data Analysis, at the Faculty of Sciences, University of Porto (2021). Population Genetics, Machine Learning, Bioinformatics.

Duarte Coelho was born in Portugal, in 1995. He got his BSc degree in Computer Engineering in 2016 from ISEP, master's degree in Computational Systems, in 2018, from ISEP. He is currently a Data Scientist at E-goi, Matosinhos, Portugal, Researcher on the Interdisciplinary Studies Research Center. His research interests involve artificial intelligence, machine learning, optimization algorithms, intelligent marketing and computational intelligence.

Fábio Fernandes was born in Vila Nova de Famalicão, Braga, Portugal on February 12, 1986. With two postgraduate in Business intelligence & Analytics at Porto Business School, Matosinhos, Portugal, 2019. Big data in Sport at Universidad Católica de Murcia (UCAM), Murcia, Spain, 2021. One degree in Marketing Management at ISMAI, Maia, Portugal, 2016. Fields of study, Analytics, Data Science, Business Intelligence.

Ivo Pereira was born in Porto, Portugal, in 1984. He got his BSc degree in Informatics Engineering by Institute of Engineering-Polytechnic of Porto in 2006, MSc degree in Informatics Engineering by Institute of Engineering-Polytechnic of Porto in 2009 and PhD in Electronics and Computers Engineering by University of Trás-os-Montes and Alto Douro in 2014. His research interests involve artificial intelligence, machine learning, optimization algorithms, intelligent marketing and computational intelligence.