Expert Systems With Applications 146 (2020) 113149

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

Combining novelty and popularity on personalised recommendations via user profile learning

Ricardo Mitollo Bertani[a], Reinaldo A. C. Bianchi[b], Anna Helena Reali Costa[a,*]

[a] Escola Politécnica, Universidade de São Paulo, São Paulo, Brazil
[b] Department of Electrical Engineering, FEI University Centre, São Bernardo do Campo, Brazil

ARTICLE INFO

Article history:
Received 12 September 2019
Revised 22 November 2019
Accepted 18 December 2019
Available online 26 December 2019

Keywords:
Recommender systems
Machine learning
Data sparsity
Diffusion-based algorithms
User profile

ABSTRACT

Recommender systems have been widely used by large companies in the e-commerce segment as aid tools in the search for relevant contents according to the user's particular preferences. A wide variety of algorithms have been proposed in the literature aiming at improving the process of generating recommendations; in particular, a collaborative, diffusion-based hybrid algorithm has been proposed in the literature to solve the problem of sparse data, which affects the quality of recommendations. This algorithm was the basis for several others that effectively solved the sparse data problem. However, this family of algorithms does not differentiate users according to their profiles. In this paper, a new algorithm is proposed for learning the user profile and, consequently, generating personalised recommendations through diffusion, combining novelty with the popularity of items. Experiments performed in well-known datasets show that the results of the proposed algorithm outperform those from both diffusion-based hybrid algorithm and traditional collaborative filtering algorithm, in the same settings.

1. Introduction

Due to the massive amount of information produced every day by both humans and machines, it became increasingly difficult to select the most suitable content in a wide range of options. Recommender systems (RS) are being widely used to help users deal with large volumes of information available on the Internet, especially in the search for the most relevant content according to the particular preferences of the user. Due to the great popularity of this kind of system and the need to ensure that such tools offer recommendations with high quality and relevance to the user, the algorithms that generate these recommendations must be constantly improved.

According to Zhang, Yao, and Sun (2017), a RS can be considered a useful information filtering tool for users aiming at discovering products or service they might be interested in and its main applications include the recommendation of movies, songs, books, documents, websites, touristic attractions, and learning materials (Lu, Wu, Mao, Wang, & Zhang, 2015).

Kotkov, Wang, and Veijalainen (2016) and Ricci, Rokach, Shapira, and Kantor (2010) consider any content that may be recommended to the user as an "item"; this term can refer to a song, a movie, a book, a service and even friends in a social network. This article will also use the term item as a reference to any type of content recommended by RS. Also, these items may have attributes that characterize them. For example, a book can be characterized by its genre, topic or author.

As stated by, a RS can be seen as a particular case of information retrieval, where by the objective is to infer the level of relevance of a set of unknown items to a target user and to generate a list of recommendations composed of items ordered by relevance.

The process of generating a recommendation can be characterized as a combination of the following features: the type of data available for analysing user preference; the filtering algorithm considered; the approach used (whether or not based on the direct use of the data); the recommended technique used (e.g., nearest neighbour algorithms, fuzzy models, singular value decomposition, bio-inspired algorithms, among others); and the dispersion level of the data-set (Bobadilla, Ortega, Hernando, & Gutiérrez, 2013).

When applying the analysis of user preference, it is fundamental to know the level of relevance that a set of items has for a given user. This information can be obtained explicitly through a rating value, as in Javari and Jalili (2014), Liu, Hu, Mian, Tian, and Zhu (2014) and Zhang et al. (2017), or it can be implicitly obtained when the RS infers the level of relevance of the item to a user through behavioural analysis, such as counting the number of times the user clicks on-screen elements, or monitoring user downloads or searches, as in Sánchez-Moreno, Gil González,

* Corresponding author.
E-mail addresses: ricardo.bertani@usp.br (R.M. Bertani), rbianchi@fei.edu.br (R. A. C. Bianchi), anna.reali@usp.br (A.H.R. Costa).

https://doi.org/10.1016/j.eswa.2019.113149
0957-4174/

Muñoz Vicente, López Batista, and Moreno García (2016) and Lacerda (2017).

Recommender systems can be broadly categorized, mainly, as content-based (CB) or collaborative filtering (CF) (Beel, Gipp, Langer, & Breitinger, 2016; Katarya & Verma, 2016; Lu et al., 2015), which will be discussed in more detail in the following section, especially the CF approach, due to its huge popularity and relevance.

The remainder of this paper is structured as follows. In Section 2 we review some relevant related work, especially the details regarding CF approaches. Section 3 describes in detail the diffusion-based hybrid algorithm, which is the basis of our proposal. The proposed method is described in Section 4 and its experimental evaluation is presented in Section 5. Finally, Section 6 gives our conclusions and indications of future work.

2. Related work

Both CB and CF seek to identify the most interesting items according to user preferences; however, each approach performs this same process according to a particular criterion.

The underlying idea of CB is that a user is interested in items that are similar to those she or he previously liked Zhang and Zeng (2015), Wang et al. (2017) and Deng, Zhong, Lü, Xiong, and Yeung (2017). CB approaches utilize a series of discrete features of an item to recommend additional items with similar features. CB approaches work with data the user provides, either explicitly (rating) or implicitly (by clicking on a link). Based on these data, a user profile is generated, which is then used to make suggestions to the user. As the user provides more inputs or takes actions on the RS, the engine becomes more and more accurate.

In turn, the idea of CF is that a user likes items that other users like. CF approaches building a model from a user's past behaviour (items previously purchased or selected and/or numerical ratings given to those items) as well as similar decisions made by other users. This model is then used to predict items (or ratings for items) that the user may have an interest in.

Algorithm 1 describes a basic CF algorithm, detailed in Ricci et al. (2010), Patra, Launonen, Ollikainen, and Nandi (2015) and Yang, Wu, Zheng, Wang, and Lei (2016) and uses the Pearson Correlation Coefficient (PCC) (Pearson, 1920) to compute the similarity between a couple of users. PCC is used in Step 4 of Algorithm 1 and can be calculated by:

Algorithm 1 – The collaborative filtering (*CF*) Algorithm (Ricci et al., 2010), (Patra et al., 2015), (Yang et al., 2016).

Require: user u, *dataset*, $|L|$, $|Vz|$.

1: *ratingsMatrix* ← Extract the rating matrix from users and items of the *dataset*
2: **for** each user $v \in dataset$, $v \neq u$ **do**
3: I_{uv} ← Separate the items rated by both u and v in the *ratingsMatrix*
4: Calculate $PCC(u, v)$ using I_{uv} and *ratingsMatrix* (Eq. 1)
5: **end for**
6: $List_{users}$ ← Sort users decreasingly according to *PCC*
7: Vz ← Separate the first $|Vz|$ from $List_{users}$
8: I_u ← Obtain the items u interacted with
9: **for** each item $i \notin I_u$ **do**
10: Calculate $pred(u, i)$ using Vz as the user neighbourhood (Eq. 2)
11: **end for**
12: *list* ← Sort all items by $pred(u, i)$
13: L ← Separate the first $|L|$ from *list*
14: **return** L

$$PCC(u,v) = \frac{\sum_{i \in I_{uv}} (r_{u,i} - \bar{r}_u)(r_{v,i} - \bar{r}_v)}{\sqrt{\sum_{i \in I_{uv}} (r_{u,i} - \bar{r}_u)^2 \sum_{i \in I_{uv}} (r_{v,i} - \bar{r}_v)^2}}, \quad (1)$$

where I_{uv} is the set of items rated by both u and v; $r_{u,i}$ and $r_{v,i}$ are the rating values attributed to item i by users u and v, respectively; $\bar{r}_u$ and $\bar{r}_v$ are the average ratings attributed to all items which u and v have interacted with, respectively; with $PCC(u, v) \in \mathbb{R}$, $-1.0 \leq PCC(u, v) \leq 1.0$.

In step 10, Algorithm 1 is calculated (Patra et al., 2015):

$$pred(u,i) = \bar{r_u} + \frac{\sum_{b \in Vz} PCC(u,b) * (r_{b,i} - \bar{r_b})}{\sum_{b \in Vz} PCC(u,b)}, \quad (2)$$

where $\bar{r_u}$ is the rating average attributed to items which user u has interacted with; Vz is the set of users similar to u (neighborhood); $PCC(u, b)$ is the similarity between u and b; $r_{b,i}$ is the rating attributed to item i by user b; and $\bar{r_b}$ is the average rating attributed to all items with which user b has had some interaction. According to Lu, Shambour, Xu, Lin, and Zhang (2013) and Kaminskas and Bridge (2016), values of $|Vz|$ between 20 and 40 are considered those that maximize the recommendation results.

CF algorithms stand out as the most widely used (Fu, Qu, Moges, & Lu, 2018; Wang et al., 2017; Yang et al., 2016). However, they still suffer from the sparsity data problem, characterized by the situation in which there are a large number of users and items in the system, but a small set of ratings assigned to items by users. Thus, the user-item rating matrix is very sparse (Ma et al., 2015).

The basic ideas of CB and CF approaches are rather vague and leave room for different approaches (Beel et al., 2016; Betru & Onana, 2017; Deng et al., 2017; Katarya & Verma, 2016; Lu et al., 2015; Wang et al., 2017; Zeng, Zeng, Shang, & Zhang, 2013; Zhang et al., 2017; Zhou et al., 2010).

Notably, proposals based on the physical processes of mass diffusion showed significant results to mitigate this particular problem (Deng et al., 2017; Wang et al., 2017; Zeng et al., 2013; Zhang & Zeng, 2015; Zhou et al., 2010). Particularly, the hybrid algorithm presented in Zhou et al. (2010) served as the basis for developing several approaches, such as Zeng et al. (2013), Zhang and Zeng (2015) and Deng et al. (2017). However, this algorithm and its variations do not differentiate the particular preferences of users during the recommendation process.

We are here particularly interested in determining the degree of novelty or popularity that represents the preferences of a particular target user. This personalised preference is expressed in the user profile that commands the decisions of the new RS algorithm we propose, the *UPOD* algorithm – "User Profile-Oriented Diffusion". Thus, according to user preference, a certain level of combination of novelty and popularity is defined, directing the selection and ordering recommended items. Our proposal experimentally demonstrated that user satisfaction was higher with the list of recommendations generated by *UPOD* than with the list generated with commonly used techniques.

It is important to emphasize that without the inclusion of the user profile during the recommendation process, the system cannot be able to fully satisfy users regarding their preferences and can simply recommend the most predictable content.

3. Diffusion-based recommendation algorithms

This section presents the hybrid algorithm proposed by Zhou et al. (2010), which is based on the physical mass diffusion process and gives a solution to the apparent diversity-accuracy dilemma of the recommender systems. The concern in this dilemma is to find an appropriate combination of methods based on accuracy (providing popularity) and diversity (providing

novelty). Their algorithm, called *MDHS*, combines the Mass Diffusion algorithm (*MD*) (Zhou, Ren, Medo, & Zhang, 2007) and the Heat Spreading algorithm (*HS*) (Zhang, Blattner, & Yu, 2007) for generating a list of recommendations.

MDHS represents an RS system as a user-item bipartite graph, formally defined as $G = \{U, I, E\}$, where $U = \{u_1, u_2, \ldots, u_N\}$, $I = \{i_1, i_2, \ldots, i_M\}$ and $E = \{e_1, e_2, \ldots, e_K\}$ are the user set, the item set and the set of graph edges, respectively. From a previously known training data-set, graph G is constructed, assigning each user to a vertex $u_N \in U$, each item to a vertex $i_M \in I$, and if a user u_N interacted with any item i_M, an edge $e_K \in E$ is inserted into G, making u_N adjacent to i_M. Interaction here means that the user purchased or rated a given item, for example.

For a target user u that is present in the database, the following three steps are performed:

- **Step 1.** A resource value $r(u, i)$ is assigned to each item i in G, according to the following rule: if there is an edge between i and the target user, then $r(u, i) = 1$, otherwise $r(u, i) = 0$. In this Step u is the target user.
- **Step 2.** The resource values are redistributed, through a propagation process, from item-side to the user side in G, where each user $v \in U$ at a destination vertex receives a new recalculated resource value $r'(v, i)$ from all the resource values of adjacent items in graph G.
- **Step 3.** The resource values assigned to the users are now redistributed from the user side to the item side in G. Each item i receives a new resource value $r''(v, i)$ recalculated from all the resource values of adjacent users, computed in Step 2.

The calculation of the new resource values described in Steps 2 and 3, and represented respectively by $r'(v, i)$ and $r''(v, i)$, depends on the algorithm considered, *MD, HS* or *MDHS*, and the number of interactions between users and items in the RS database. The number of interactions represents the degree of vertices, $w(u)$ and $w(i)$, being respectively the degree of the vertex representing user u and the degree of the vertex representing item i.

Thus, when considering only the *MD* algorithm, the resource values for steps 2 and 3 are given, respectively, by:

$$r'_{MD}(v, i) = \sum_{i \in I} \frac{r(u, i)}{w(i)}, v \in U, i \in I, \tag{3}$$

and

$$r''_{MD}(v, i) = \sum_{v \in U} \frac{r'_{MD}(v, i)}{w(v)}, v \in U, i \in I. \tag{4}$$

Similarly, when only the *HS* algorithm is used, the resource values for steps 2 and 3 are given, respectively, by:

$$r'_{HS}(v, i) = \frac{\sum_{i \in I} r(u, i)}{w(v)}, v \in U, i \in I, \tag{5}$$

and

$$r''_{HS}(v, i) = \frac{\sum_{v \in U} r'_{HS}(v, i)}{w(i)}, v \in U, i \in I. \tag{6}$$

Figs. 1 and 2 illustrate the application of the propagation process (steps 1, 2 and 3) in a graph considering the *MD* and *HS* algorithms, respectively.

In Fig. 1, the target user is $u4$, which previously interacted with items $i3$ and $i5$. Following the MD algorithm, resource values are propagated to the user side using Eq. (3); in this case, $i3$ propagates 1/2 to adjacent users in the graph and $i5$ propagates 1/3, resulting in 5/6 at $u4$. In the last step, the resources are propagated to the side of the items according to Eq. (4); the new value in item $i3$, for example, is 5/12 propagated from $u4$ added with 1/6 propagated from $u1$, resulting in 7/12.

Fig. 2 illustrates the propagation of the HS algorithm, following Eqs. (5) and (6). In the figure, the target user is $u4$, which previously interacted with items $i3$ and $i5$. Resource values are then propagated from the item to the user side, resulting in 1/3 in $u1$ (sum of the resources received from $i1$, $i2$, and $i3$, divided by the degree of $u1$), and 1/2 in $u2$ (sum of the resources received from $i2$ and $i5$, divided by the degree of $u2$). In the last step, the resources are propagated from the users to the items side according to Eq. (6); the new value in item $i3$, for example, is 2/3 (sum of 1/3 propagated from $u1$ added with 1 propagated from $u4$, divided by degree of $i3$, which is 2).

It is worth observing that, according to Figs. 1 and 2, the *MD* algorithm tends to generate recommendations composed of popular items (vertex with the higher degrees in the graph), while the *HS* algorithm tends to generate recommendations composed of the lesser-known items (vertex with lower degrees in the graph).

The *MDHS* algorithm allows combining *MD* and *HS* by using a tuning parameter λ used to combine Eqs. (3) and (5) into $r'_{HB}(v, i)$ (Step 2) and Eqs. (4) and (6) in $r''_{HB}(v, i)$ (Step 3):

$$r'_{HB}(v, i) = \left(\sum_{i \in I} \frac{r(u, i)}{w(i)^{\lambda}} \right) \Big/ w(v)^{(1-\lambda)}, v \in U, i \in I, \tag{7}$$

and

$$r''_{HB}(v, i) = \left(\sum_{v \in U} \frac{r'_{HB}(v, i)}{w(v)^{\lambda}} \right) \Big/ w(i)^{(1-\lambda)}, v \in U, i \in I. \tag{8}$$

When $\lambda = 0$ the *HS* algorithm is used, and when $\lambda = 1$ the *MD* algorithm is used. Any value between 0 and 1 means that a combination of both methods is used. In Zhou et al. (2010), the values of λ are defined as $0 \leq \lambda \leq 1$, and they recommend using $\lambda = 0.5$. After the execution of the three steps described before, any item that, at the beginning of the process, did not have any interaction with target user u and that has received a positive resource value ($r''(u, i) > 0$), is sorted by its final resource value $r''(u, i)$ in ascending order and included in the prediction list.

The items with the highest resource values in the prediction list will compose the recommendation list for target user u. The list of recommendations consists of $|L|$ items and is sorted in descending order by the item resource values; the first position contains the most relevant item and the last one contains the least relevant item.

Some extensions of *MDHS* have been proposed in the RS literature: the Semi-Local Diffusion algorithm (*SLD*) (Zeng et al., 2013) allows the propagation process to be repeated in G. In practice, this means that steps 2 and 3 occur more than once, with recommendations reaching farther user-item neighbourhoods.

However, both *MDHS* and *SLD* algorithms always consider the same λ value for all RS users, without considering any information about their profiles. This means that for each value of λ, recommendation lists are generated with the same degree of combination of *MD* and *HS* for all users.

We believe that using the same value of λ for all RS target users, without differentiating them according to their specific profiles, leads to worse results. We here propose a framework that tunes the λ parameter according to the user profile, and we combine this λ parameter with the *MDHS* algorithm to generate personalised recommendations. Our proposal is described in the next section.

4. Proposal

We propose the *UPOD* algorithm – "User Profile Oriented Diffusion". In the *MDHS* algorithm, *UPOD* uses a λ value especially learned for the target user in order to generate customized recommendations for the target user.

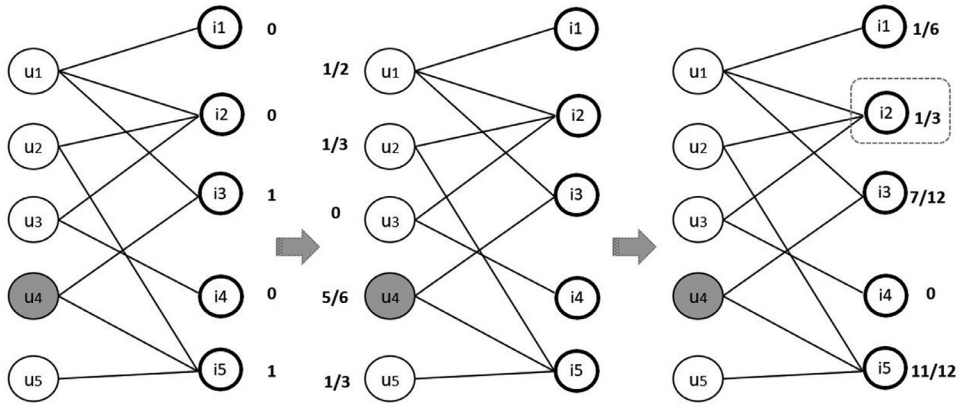

Fig. 1. The three propagation steps with the *MD* algorithm in a user-item bipartite graph. Users are represented by circles, items are represented by bold circles, the target user is indicated by a shaded circle and the most relevant item to be recommended is marked with a dotted rectangle (modified figure from Zhou et al. (2010)).

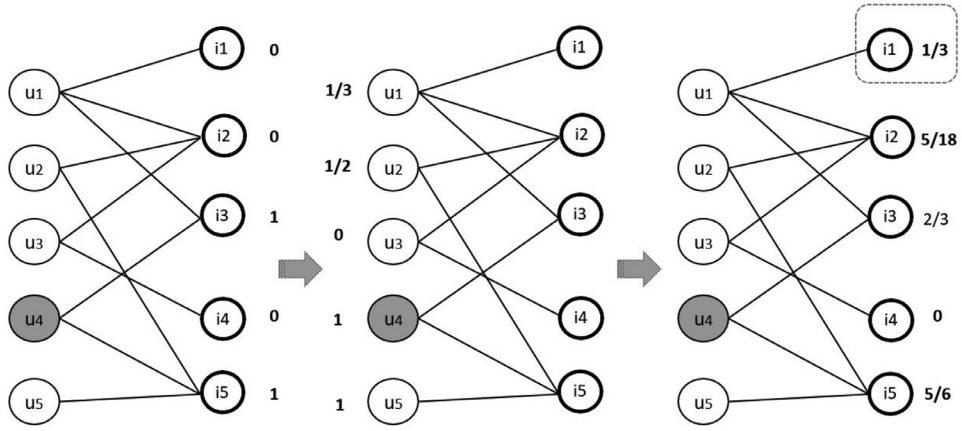

Fig. 2. The three propagation steps with the *HS* algorithm in a user-item bipartite graph. Users are represented by circles, items are represented by bold circles, the target user is indicated by a shaded circle and the most relevant item to be recommended is marked with a dotted rectangle (modified figure from Zhou et al. (2010)).

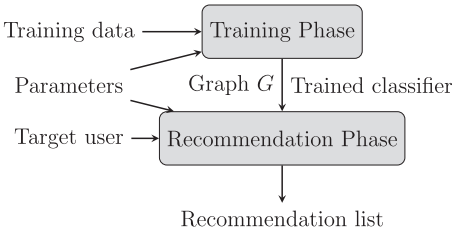

Fig. 3. The two phases of the *UPOD* algorithm.

UPOD operates in two phases: a training phase and a recommendation phase, as shown in Fig. 3.

The training phase is responsible for preprocessing the data, defining the features of the users, constructing the bipartite graph of interactions, clustering users according to features, determining the profile of each cluster from the λ that characterizes the cluster and, finally, using the pair that defines the user features and the best λ for each cluster. The training phase trains a classifier that will provide the best λ from the features of a target user. This procedure is illustrated in Algorithm 2.

The training phase is initially responsible for preprocessing the training *dataset* (Step 1), by deleting users with invalid or empty data, removing the timestamp, transforming the zip code into a country and state format, etc. Also in this step, attribute values are all transformed into categorical values. For example, the user age attribute is discretized every 5 years and each interval represents one category.

From Step 2 to 11, the preprocessed data are grouped into k clusters by performing the *k-modes* algorithm (Huang, 1998). The *k-modes* algorithm uses a dissimilarity measure for categorical data, represents cluster centroids with modes, and uses a frequency-based method to update modes in the clustering process. This algorithm was chosen due to its simplicity, easy understanding and implementation, and its efficiency, which is O(tkn), where n is the number of data, k is the number of clusters and t is the number of iterations. Since k and t are small, *k-modes* is considered a linear algorithm and it is a variant of k-means for categorical data.

Let X and Y be attribute vectors that represent categorical data, where $x_j \in X$ and $y_j \in Y$ are the values of each data attribute. The measure of dissimilarity between X and Y in *k-modes* is given by:

$$d(X,Y) = \sum_{j=1}^{m} \delta(x_j, y_j), \tag{9}$$

where $\delta(x_j, y_j)$ is equal to 0 if $x_j = y_j$ and is equal to 1 if $x_j \neq y_j$.

Consider an example in which vectors have 3 attributes: a1 = gender, a2 = profession, and a3 = age group. An X vector could be x1 = male, x2 = teacher and x3 = 15–20 and Y could be y1 = female, y2 = teacher and y3 = 20–25. In this case, the measure of dissimilarity d(X,Y) = 1 + 0 + 1, indicating that of the 3 attributes, only 1 is identical in both vectors.

A mode of a set of n categorical objects X_i is a vector **Q** (a categorical attribute vector) that has the least dissimilarity to all vectors (elements) in a given cluster and therefore minimizes the

Algorithm 2 – *UPOD* training phase.

Require: *dataset*, *Feat*, *kmin*, *kmax*, Λ, *Metric*
1: $TrainData \leftarrow prep(dataset)$
2: $Eval \leftarrow 0$
3: **for** k = *kmin* to k = *kmax* **do**
4: $Clusters_k \leftarrow clustering(TrainData, Feat, k)$
5: $Eval_k \leftarrow evaluate(Clusters_k)$
6: **if** $Eval_k$ is better than *Eval* **then**
7: $Eval \leftarrow Eval_k$
8: $BestClusters \leftarrow Clusters_k$
9: $Best_k \leftarrow k$
10: **end if**
11: **end for**
12: $G \leftarrow buildGraph(TrainData)$
13: $TrainSet \leftarrow \{\}$
14: **for** $i = 1$ to $i = Best_k$ **do**
15: $\lambda_{best}[i] \leftarrow BestLambda(G, BestClusters[i], \Lambda, Metric)$
16: **for** each user in BestClusters[i] **do**
17: $userFeat \leftarrow ExtractFeat(user, Feat)$
18: $TrainSet \leftarrow TrainSet \cup \{< userFeat, \lambda_{best}[i] >\}$
19: **end for**
20: **end for**
21: $classifier \leftarrow TrainClassifier(TrainSet)$
22: **return** *classifier*, *G*

function $D(X, \mathbf{Q})$:

$$D(X, \mathbf{Q}) = \sum_{i=1}^{n} d(X_i, \mathbf{Q}). \tag{10}$$

Thus, in Step 4 of Algorithm 2, the *clustering* function represents the *k-modes* algorithm, used with *k* varying from *kmin* to *kmax* and, for each *k*, an evaluation of the *k* clusters is performed. The *k-modes* algorithm is shown in Algorithm 3 (Huang, 1998).

Algorithm 3 – The *k-modes* Algorithm.

Require: *TrainData*, *Feat*, *k*
Select *k* initial modes, one for each cluster
for each object *o* from $< TrainData, Feat >$ **do**
Allocate *o* to the cluster whose mode is the nearest to it according to Eq.9
end for
for each mode from each cluster **do**
Retest the dissimilarity of objects against the current modes
if its nearest mode belongs to another cluster **then**
Reallocate the object to that cluster
Update the modes of both clusters
end if
end for
$Clusters_k \leftarrow$ set of *k* final clusters
return $Clusters_k$

In Step 5 of Algorithm 2, for each *k* cluster returned by the clustering function that implements the *k-modes* algorithm (Algorithm 3), we evaluate the effectiveness of the data partitioning achieved. For this, we use the entropy-based clustering criterion given by Chen and Liu (2009). Consider a dataset $X = \{x_1, x_2, \ldots, x_n\}$ with *n* instances and *c* columns. Each instance x_i is described by *c* columns ($x_i = \{x_1^i, x_2^i, \ldots, x_c^i\}$) and each column of x_i has a value from domain A_i, in which there are a finite number of unique categorical values. If $v \in A_i$, then the probability of $x_i = v$ is given by $p(x_i = v)$ and the column entropy of A_i is given by:

$$H(A_i \mid X) = -\sum_{v \in A_i} p(v \mid X) \log_2 p(v \mid X), \tag{11}$$

where $p(v|X)$ is an empirical probability estimated in *X*. The estimated entropy of the whole dataset is the sum of all the columns entropy and is represented by $H(X)$. Besides, supposing dataset *X* is partitioned into *k* clusters, being $C^k = \{C_1, C_2, \ldots C_k\}$ the set of clusters (partition), C_k is a cluster, n_k is the number of objects in C_k and $H(C_k)$ is the cluster entropy. According to Chen and Liu (2009), the entropy-based clustering criterion is given by

$$Opt(C^k) = \frac{1}{c}\left(H(X) - \frac{1}{n}\sum_{k=1}^{k} n_k H(C_k)\right). \tag{12}$$

Notice that $H(X)$ is fixed and depends on the dataset, so that the maximization of $Opt(C^k)$ is equivalent to minimizing the expression $\frac{1}{n}\sum_{k=1}^{k} n_k H(C_k)$, which is named expected entropy from partition C^k. Chen and Liu (2009) argue that the "best k" is the one that results in the minimum expected entropy among all *k* clusters. Hence, at the end of this loop (Steps 3 – 11), we have the best set of clusters for the best *k* in *BestClusters*, given in $Best_k$.

Now that we have the best set of dataset clusters, we need to determine for each cluster the best λ, λ_{best}, which represents the profile of the users included in that cluster. This is done in Step 15 of Algorithm 2, with algorithm *BestLambda* given in Algorithm 4.

Algorithm 4 – The *BestLambda* Algorithm.

Require: *G*, *BestCluster*[*i*], *Metric*, Λ
1: $Eval_\lambda \leftarrow 0$
2: **for** each λ value in Λ **do**
3: $RecList_\lambda \leftarrow MDHS(G, BestCluster[i], \lambda)$
4: $Eval \leftarrow Metric(RecList_\lambda)$
5: **if** *Eval* is better than $Eval_\lambda$ **then**
6: $Eval_\lambda \leftarrow Eval$
7: $\lambda_{best}[i] \leftarrow \lambda$
8: **end if**
9: **end for**
10: **return** $\lambda_{best}[i]$

In this algorithm, each recommendation list generated with each λ is evaluated (Step 4) according to metrics, such as Ranking Score, Precision and Recall (Ma, Ren, Wu, Wang, & Feng, 2017; Zeng, An, Liu, Shang, & Zhou, 2014), to determine the λ that characterizes the best-generated recommendation list; this λ will be λ_{best}. Note that in our system, λ_{best} represents the appropriate combination of novelty and popularity for recommendations according to the user in question and Λ set contains all the possible lambda values considered by the authors of *MDHS* in the base article, being: $\Lambda = \{0.0, 0.1, 0.2, 0.3, 0.4, 0, 5, 0.6, 0.7, 0.8, 0.9, 1.0\}$.

Once the $\lambda_{best}[i]$ for that cluster *i* in the set *BestClusters* has been determined, we collect the features of each user included in that cluster to compose a training pair for the classifier (see Algorithm 2). This pair, $< userFeat, \lambda_{best}[i] >$, is inserted into the classifier training dataset, *TrainSet* (Steps 17 and 18). Finally, using the training data *TrainSet*, a classifier is trained in Step 21 so that, given features such as age, gender, and occupation of a given target user, the classifier can indicate which the best λ is that represents that user profile. The output of the training phase is the trained classifier and graph *G* of interactions between users and items in the original dataset. For the results presented here, we use an SVM (Support Vector Machine) classifier (Witten et al., 1999). However, any other kind of classifier could be used.

The classifier is used to predict the best value of λ for a target user in the recommendation phase. The recommendation phase requires as input the interaction graph *G*, the set of user attributes *Feat*, the target user, the classifier trained in training phase and a size for the recommendation list *L*. The *UPOD* Recommendation phase is illustrated in Algorithm 5.

Algorithm 5 – *UPOD* Recommendation phase.

Require: G, $Feat$, $user$, $classifier$, $|L|$.
1: $userFeatures \leftarrow ExtractFeat(user, Feat)$
2: $\lambda_{user} \leftarrow classifier(userFeatures)$
3: $recommendList \leftarrow MDHS(G, user, \lambda_{user}, |L|)$
4: **return** $recommendList$

Table 1
Sparsity level of the used dataset.

Dataset	Sparsity
MovieLens	0.9371
Last.FM	0.9989
Book-Crossing	0.9996

First, the target user's feature values, *userFeatures*, are extracted in Step 1 considering the features defined in the set *Feat* and the values indicated by the target user. Values represented in *userFeatures* are inputs to the trained classifier, which predicts the proper λ value for the user, represented by λ_{user} at Step 2. This variable λ_{user} reflects the target user profile, defining the appropriate combination of the MD algorithm, which selects popular items, with the HS algorithm, which selects items based on novelties. Then, in Step 3 the *MDHS* algorithm is executed on graph G for the target user and his/her personalised tuning parameter (λ_{user}) and generates the recommendation list of size $|L|$. Note that now the generated recommendation list combines popular and lesser-known items according to the particular user profile.

5. Experiments and results

We performed a series of experiments to validate the proposed algorithm. For this, we performed a comparative analysis with two recommender systems and three different databases. We evaluate the results according to some metrics that are widely used in this type of application. The details of the experimental procedures and the results are described below.

5.1. Data bases

We considered three well-known datasets in the RS literature. The datasets used are:

- **The MovieLens dataset** (Harper & Konstan, 2015) contains 910 users, 1672 items and 95,579 interactions. The user features are age, gender and location.
- **The Last.FM dataset** (Celma, 2010). This work used a random subset of the total dataset, containing 2846 users, 4995 items and 14,583 interactions. The user features are age, gender and country.
- **The Book-Crossing dataset** (Ziegler, McNee, Konstan, & Lausen, 2005) is also a random subset of the total dataset, containing 3421 users, 26,811 items and 35,572 interactions. The user features are age and location.

According to Lu et al. (2013) and Shambour and Lu (2015), the sparseness of a given dataset can be calculated by

$$sparsity = 1 - \frac{K}{NM}, \tag{13}$$

where K, N and M are, respectively, the number of interactions between users and items, the number of users and the number of items.

Using Eq. (13), we have computed the sparsity level of MovieLens, Last.FM and Book-Crossing datasets shown in Table 1. The Book-Crossing dataset is the sparsest.

5.2. Algorithms for comparison

We have comparatively evaluated the proposed algorithm with two other algorithms: (i) the *MDHS* algorithm with $\lambda = 0.5$, following the author's suggestion, and (ii) a nearest neighbourhood collaborative filtering (CF) algorithm, which was previously detailed in Algorithm 1 in Section 1.

5.3. Evaluation method and metrics

We used the 10-fold cross-validation technique to validate our proposal. Cross-validation is a model validation technique for assessing how the results of a statistical analysis will generalize to an independent dataset. This approach involves randomly dividing the set of data into k groups, or folds, of approximately equal size. The first fold is treated as a testing set, and the method is fit on the remaining $k-1$ folds (called the training set). The procedure is repeated for each group, and the validation results are averaged over the rounds to give an estimate of the model predictive performance. We only consider users that are present in both the training set and the testing set. These users are called "valid users" and are represented by U_{valid}.

We used three metrics to validate the recommendation list generated by the systems: Recall, Precision, and Ranking Score (Ma et al., 2017; Zeng et al., 2014).

Recall: It measures the proportion of items in the testing set that correspond to items in the recommendation list generated for a target user u (Yang et al., 2016):

$$Re_u(L) = \frac{d_u(L)}{I_u}, \tag{14}$$

where $d_u(L)$ is the number of items related to target user $u \in U_{valid}$ in the testing set and also present in recommendation list L; I_u is the total number of items related to $u \in U_{valid}$ in the testing set.
The higher its value, the more items in the testing set correspond to recommended items.

Precision: It measures the proportion of items in the user recommendation list that corresponds to items related to the target user in the testing set, and is given by (Yang et al., 2016):

$$Pe_u(L) = \frac{d_u(L)}{|L|}, \tag{15}$$

where $d_u(L)$ is the number of items related to the target user $u \in U_{valid}$ in the testing set that are present in the recommendation list and $|L|$ is the size of the recommendation list. The higher its value, the more items in the recommendation list correspond to items related to the user in the testing set.

Ranking Score: It takes into account the position in which items related to the target user in the testing set appear in the generated recommendation list. This position is known as the "rank" of the item, which is obtained based on the position of the item in the recommendation list divided by the number of items initially unknown to the user (Chen, Zeng, & Chen, 2015; Yu, Zeng, Gillard, & Medo, 2015):

$$RS_u = \sum_{i \in I_u} \frac{rank(i)}{|I_u|}, \tag{16}$$

where I_u is the set of items related to $u \in U_{valid}$ in the testing set and $rank(i)$ represents the rank of item i in the generated recommendation list. This means that the closer to the first positions the item is, the lower the rank value.

Table 2
Experimental results for the three datasets.

Algorithm	RS		Re(L)		Pe(L)	
	avg	σ	avg	σ	avg	σ
CF	0.497	0.002	0.014	0.002	0.003	$4.6x10^{-4}$
MDHS ($\lambda = 0.5$)	0.080	0.001	0.299	0.008	0.086	0.001
UPOD	0.074	0.001	0.308	0.007	0.091	0.002

(a) Results from the **Movielens** dataset. With $k_{avg} = 197$ and $k_{\sigma} = 4.25$ for the *UPOD* algorithm. The average processing time for CF, MDHS and UPOD were 5 min, 30 s and 5 s, respectively.

Algorithm	RS		Re(L)		Pe(L)	
	avg	σ	avg	σ	avg	σ
CF	0.444	0.004	0.018	0.004	$9.6x10^{-4}$	$2x10^{-4}$
MDHS ($\lambda = 0.5$)	0.252	0.007	0.119	0.006	0.006	$2.9x10^{-4}$
UPOD	0.251	0.006	**0.174**	**0.014**	**0.009**	$\mathbf{7.4x10^{-4}}$

(b) Results from the **Last.FM** dataset. With $k_{avg} = 196$ and $k_{\sigma} = 2.94$ for the *UPOD* algorithm. The average processing time for CF, MDHS and UPOD were 2 min, 18 s and 9 s, respectively

Algorithm	RS		Re(L)		Pe(L)	
	avg	σ	avg	σ	avg	σ
CF	0.406	0.007	$3.6x10^{-4}$	$2.4x10^{-4}$	$3.5x10^{-5}$	$1x10^{-5}$
MDHS ($\lambda = 0.5$)	0.395	0.007	0.002	$4.7x10^{-4}$	$2x10^{-4}$	$4.9x10^{-5}$
UPOD	0.394	0.008	**0.003**	**0.001**	$\mathbf{3.76x10^{-4}}$	$\mathbf{7.3x10^{-5}}$

(c) Results from the **Book-Crossing** dataset. With $k_{avg} = 192$ and $k_{\sigma} = 8.5$ for the *UPOD* algorithm. The average processing time for CF, MDHS and UPOD were 20 min, 30 s and 5 s, respectively.

5.4. Implementation details

An important aspect to consider is how the proposed new algorithm, UPOD, can be compared with the other two algorithms in terms of time efficiency. To explain how we can measure this efficiency, it is first necessary to understand some implementation details involved in the structure, as well as the computational conditions under which the algorithms were executed.

One of the most important differences between the UPOD and MDHS implementations is that the first one performs the propagation steps using a parallel computing strategy. This means that, in our implementation, the propagation steps occur for more than one user at a time, depending on the number of processor cores, unlike the original MDHS implementation, which performs propagations for each user, one after the other.

All the algorithms were executed on a Debian GNU Linux with 7.5 GB RAM and 2 vCPUs available on the Google Cloud Platform (Google Compute Engine) and all the implementations were made with the *Java*^{TM} programming language. Under these conditions, the UPOD training phase time was 3 min for the Movielens database, 41 s for Last.FM and 2 min and 55 s for Book-Crossing. Of course, this may vary depending on the computational structure used and the size of the dataset, the number of user attributes, and the number of graph edges.

Note that the training phase takes place offline and only when needed.

5.5. Results and discussion

For all the experiments, we considered $|L| = 30$ as the size of the recommendation list. In the CF algorithm, the neighbourhood size was $|Vz| = 30$; and for the *UPOD* algorithm, $k_{\min}$ was defined experimentally as 100, while $k_{\max}$ was defined in order to maintain a maximum of 15–20 users in each cluster, so that $k_{\max} = 200$ for all datasets.

Table 2 shows the results from each dataset. Bold are those results in which *UPOD* has outperformed the CF and *MDHS* algorithms at the same time, according to the *t*-test (95% confidence interval), meaning that the difference between the results is considered to be statistically significant. k_{avg} and k_{σ} were calculated for each dataset, after the application of the entropy-based clustering criterion to each of the 10 executions.

Table 2 (a) presents the results for the **Movielens** dataset, Table 2(b) presents the results for the **Last.FM** dataset, and Table 2(c) presents the results for the **Book-Crossing** dataset.

Table 2 allows observing that *UPOD* significantly outperformed the other algorithms concerning the metrics used, although the difference between the results was not considered statistically significant in all cases. For example, according to the *t*-test, *UPOD* does not outperform CF and *MDHS* in the RankScore (RS) metric in the Last.FM and Book-Crossing datasets. However, our proposal systematically outperformed the others according to Recall and Precision metrics.

The inclusion of the user profile is a very important contribution of our proposal as it allows for more refined and personalised recommendation content.

In addition, the user profile can also be obtained in ways different from that used in the UPOD system; For example, it can be derived from social network data or even from the user behaviour observed in e-commerce.

6. Conclusions and future work

This article contributed with the *UPOD* algorithm; this allowed the automatic tuning of the λ parameter, which determines the mixing degree in the mass diffusion process. *UPOD* makes it possible to generate personalised recommendations, combining popularity and novelty according to each user's particular profile. Experiments using three well-known databases have shown that supervised learning of the λ parameter according to user profile can improve the quality of recommendations in sparse data sets.

Future research should include the rating values assigned to items in the calculation of resource values during the mass diffusion process in the bipartite graph. Thus, better-rated items in the system would be recommended more strongly than poorly rated items. Another possible improvement would be to allow a greater amount of resource propagation in the graph, as in the *Semi-Local Diffusion*, Zeng et al. (2013) algorithm. Finally, several aspects must be integrated into the design of a recommender system so that it

can generate even better recommendations that are more in line with each user's profile.

Declaration of Competing Interest

None.

Credit authorship contribution statement

Ricardo Mitollo Bertani: Conceptualization, Formal analysis, Methodology, Validation, Writing - original draft. **Reinaldo A. C. Bianchi:** Writing - original draft, Formal analysis, Conceptualization. **Anna Helena Reali Costa:** Writing - original draft, Formal analysis, Conceptualization, Funding acquisition.

Acknowledgement

The authors acknowledge the support of CNPq (N. 425860/2016-7 and N. 307027/2017-1). The authors also would like to thank all the support provided by An Zeng and Wei Zeng, authors of the paper Zeng et al. (2013), with the reproduction of the *MDHS* algorithm, which was originally presented in Zhou et al. (2010). This study was financed in part by the *Coordenação de Aperfeiçoamento de Pessoal de Nível Superior* (CAPES), Brazil, Finance Code 001.

References

Beel, J., Gipp, B., Langer, S., & Breitinger, C. (2016). Research-paper recommender systems: a literature survey. *International Journal on Digital Libraries, 17*(4), 305–338. doi:10.1007/s00799-015-0156-0.

Betru, B. T., & Onana, C. A. (2017). Deep learning methods on recommender system: A survey of state-of-the-art. *International Journal of Computer Applications, 162*(10), 975–8887. doi:10.5120/ijca2017913361.

Bobadilla, J., Ortega, F., Hernando, A., & Gutiérrez, A. (2013). Recommender systems survey. *Knowledge-Based Systems, 46*, 109–132. doi:10.1016/j.knosys.2013.03.012.

Celma, O. (2010). *Music recommendation and discovery*. ISBN 978-3-642-13287-2: Springer.

Chen, B., Zeng, A., & Chen, L. (2015). The effect of heterogeneous dynamics of online users on information filtering. Physics Letters A, *379*(43–44), 2839–2844. doi:10.1016/j.physleta.2015.09.019.

Chen, K., & Liu, L. (2009). "Best K": Critical clustering structures in categorical datasets. *Knowledge and Information Systems, 20*(1), 1–33. doi:10.1007/s10115-008-0159-x.

Deng, X., Zhong, Y., Lü, L., Xiong, N., & Yeung, C. (2017). A general and effective diffusion-based recommendation scheme on coupled social networks. *Information Sciences, 417*(168), 420–434. doi:10.1016/j.ins.2017.07.021.

Fu, M., Qu, H., Moges, D., & Lu, L. (2018). Attention based collaborative filtering. *Neurocomputing, 311*, 88–98. doi:10.1016/j.neucom.2018.05.049.

Harper, F. M., & Konstan, J. A. (2015). The movielens datasets: History and context. ACM Transactions on Interactive Intelligent Systems, *5*(4), 19:1–19:19. doi:10.1145/2827872.

Huang, Z. (1998). Extensions to the k-Means algorithm for clustering large data sets with categorical values. Data Mining and Knowledge Discovery, *2*(3), 283–304.

Javari, A., & Jalili, M. (2014). A probabilistic model to resolve diversity-accuracy challenge of recommendation systems. *Knowledge and Information Systems, 44*(3), 609–627. doi:10.1007/s10115-014-0779-2.

Kaminskas, M., & Bridge, D. (2016). Diversity, serendipity, novelty, and coverage. ACM Transactions on Interactive Intelligent Systems, *7*(1), 1–42. doi:10.1145/2926720.

Katarya, R., & Verma, O. P. (2016). Recent developments in affective recommender systems. *Physica A: Statistical Mechanics and its Applications, 461*, 182–190. doi:10.1016/j.physa.2016.05.046.

Kotkov, D., Wang, S., & Veijalainen, J. (2016). A survey of serendipity in recommender systems. *Knowledge-Based Systems, 111*, 180–192. doi:10.1016/j.knosys.2016.08.014.

Lacerda, A. (2017). Multi-Objective ranked bandits for recommender systems. Neurocomputing, *246*, 12–24. doi:10.1016/j.neucom.2016.12.076.

Liu, H., Hu, Z., Mian, A., Tian, H., & Zhu, X. (2014). A new user similarity model to improve the accuracy of collaborative filtering. *Knowledge-Based Systems, 56*, 156–166. doi:10.1016/j.knosys.2013.11.006.

Lu, J., Shambour, Q., Xu, Y., Lin, Q., & Zhang, G. (2013). A web-based personalized business partner recommendation system using fuzzy semantic techniques. *Computational Intelligence, 29*(1), 37–69. doi:10.1111/j.1467-8640.2012.00427.x.

Lu, J., Wu, D., Mao, M., Wang, W., & Zhang, G. (2015). Recommender system application developments: A survey. *Decision Support Systems, 74*, 12–32. doi:10.1016/j.dss.2015.03.008.

Ma, T., Zhou, J., Tang, M., Tian, Y., Al-Dhelaan, A., Al-Rodhaan, M., & Lee, S. (2015). Social network and tag sources based augmenting collaborative recommender system. IEICE Transactions on Information and Systems, *98*(4), 902–910.

Ma, W., Ren, C., Wu, Y., Wang, S., & Feng, X. (2017). Personalized recommendation via unbalance full-connectivity inference. *Physica A: Statistical Mechanics and its Applications, 483*, 273–279. doi:10.1016/j.physa.2017.04.041.

Patra, B. K., Launonen, R., Ollikainen, V., & Nandi, S. (2015). A new similarity measure using Bhattacharyya coefficient for collaborative filtering in sparse data. *Knowledge-Based Systems, 82*, 163–177. doi:10.1016/j.knosys.2015.03.001.

Pearson, K. (1920). Notes on the history of correlation. Biometrika, *13*(1), 25–45. doi:10.1093/biomet/13.1.25.

Ricci, F., Rokach, L., Shapira, B., & Kantor, P. B. (2010). *Recommender systems handbook* (1st). New York, NY, USA: Springer-Verlag New York, Inc..

Shambour, Q., & Lu, J. (2015). An effective recommender system by unifying user and item trust information for B2B applications. *Journal of Computer and System Sciences, 81*(7), 1110–1126. doi:10.1016/j.jcss.2014.12.029.

Sánchez-Moreno, D., Gil González, A. B., Muñoz Vicente, M. D., López Batista, V. F., & Moreno García, M. N. (2016). A collaborative filtering method for music recommendation using playing coefficients for artists and users. *Expert Systems with Applications, 66*, 1339–1351. doi:10.1016/j.eswa.2016.09.019.

Wang, X., Liu, Y., Zhang, G., Zhang, Y., Chen, H., & Lu, J. (2017). Mixed similarity diffusion for recommendation on bipartite networks. IEEE Access, *5*, 21029–21038. doi:10.1109/ACCESS.2017.2753818.

Witten, I. H., Frank, E., Trigg, L., Hall, M., Holmes, G., & Cunningham, S. J. (1999). Weka: Practical machine learning tools and techniques with Java implementations. In N. Kasabov, & K. Ko (Eds.), *Proceedings of the ICONIP/ANZIIS/ANNES'99 workshop on emerging knowledge engineering and connectionist-based information systems* (pp. 192–196). Dunedin, New Zealand

Yang, Z., Wu, B., Zheng, K., Wang, X., & Lei, L. (2016). A survey of collaborative filtering-based recommender systems for mobile internet applications. IEEE Access, *4*, 3273–3287. doi:10.1109/ACCESS.2016.2573314.

Yu, F., Zeng, A., Gillard, S., & Medo, M. (2015). Network-based recommendation algorithms: A review. *Physica A: Statistical Mechanics and its Applications, 452*, 192–208. arXiv:1511.06252. doi:10.1016/j.physa.2016.02.021.

Zeng, W., An, Z., Liu, H., Shang, M.-s., & Zhou, T. (2014). Uncovering the information core in recommender systems. *Scientific Reports, 4*, 6140. doi:10.1038/srep06140.

Zeng, W., Zeng, A., Shang, M. S., & Zhang, Y. C. (2013). Information filtering in sparse online systems: Recommendation via semi-local diffusion. *PLoS ONE, 8*(11). doi:10.1371/journal.pone.0079354.

Zhang, F.-G., & Zeng, A. (2015). Information filtering via heterogeneous diffusion in online bipartite networks. PLoS ONE, *10*(6), e0129459. doi:10.1371/journal.pone.0129459.

Zhang, S., Yao, L., & Sun, A. (2017). Deep learning based recommender system: A Survey and new perspectives. ACM Journal on Computing and Cultural Heritage (JOCCH), *1*(1), 1–35. arXiv:1707.07435.

Zhang, Y.-C., Blattner, M., & Yu, Y.-K. (2007). Publisher'S note: heat conduction process on community networks as a recommendation model. Physical Review Letters, *99*(16), 169902. arXiv:0803.2179. doi:10.1103/PhysRevLett.99.169902.

Zhou, T., Kuscsik, Z., Liu, J.-G., Medo, M., Wakeling, J. R., & Zhang, Y.-C. (2010). Solving the apparent diversity-accuracy dilemma of recommender systems.. *Proceedings of the National Academy of Sciences of the United States of America, 107*(10), 4511–4515. arXiv:0808.2670. doi:10.1073/pnas.1000488107.

Zhou, T., Ren, J., Medo, M., & Zhang, Y. C. (2007). Bipartite network projection and personal recommendation. *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics, 76*(4), 1–7. doi:10.1103/PhysRevE.76.046115.

Ziegler, C.-N. C., McNee, S. M. S., Konstan, J. a. J., & Lausen, G. (2005). Improving recommendation lists through topic diversification. In *Proceedings of the 14th international conference on World Wide Web WWW 05* (p. 22). doi:10.1145/1060745.1060754.