Multimedia Tools and Applications (2021) 81:2467–2488
https://doi.org/10.1007/s11042-021-11564-x

User encoding for clustering in very sparse recommender systems tasks

Pablo Pérez-Núnez[1] · Jorge Díez[1] · Oscar Luaces[1] · Antonio Bahamonde[1]

Received: 15 September 2020 / Revised: 14 September 2021 / Accepted: 20 September 2021 /
Published online: 29 October 2021

Abstract

Recommender Systems are a very useful tool which let companies and service providers focus in the preferences of their customers, helping them to avoid an overwhelming variety of choices. In this context, clustering tools can play an important role to detect groups of customers with similar tastes. Thus, companies can make personalized marketing campaigns, offering to their users new products which have been consumed by other users with comparable preferences. In this paper we present a general framework to cluster users with respect to their tastes when the registers stored about the interactions between users and products are extremely scarce. Commonly, clustering methods employ the values of features describing the samples to be clustered (users in our case), but such features are not always available. We propose some alternative representations for users, in which their tastes are gathered to some extent, so that clustering algorithms can take advantage and make more homogeneous groups in this regard. To illustrate the performance of the whole framework, we tested it on six popular datasets commonly used as a benchmark for recommender systems, as well as on an extremely sparse real-world dataset that records the preferences of readers to click promoted links in digital publications. In the experimental section we compare our proposed representations to other common user encodings. We show that clustering users attending only to their feature values or to the items they have evaluated gives rise to the worst scores in terms of taste homogeneity.

Keywords Recommender systems · Preference learning · Matrix factorization · Clustering · Scalability · User encoding

✉ Jorge Díez
jdiez@uniovi.es

Pablo Pérez-Núnez
pabloperez@uniovi.es

Oscar Luaces
oluaces@uniovi.es

Antonio Bahamonde
abahamonde@uniovi.es

[1] Artificial Intelligence Center, University of Oviedo, 33204 Gijón, Spain

1 Introduction

Recommender Systems (RS) is a very active research field in Machine Learning. The objective is to find patterns in the interaction between items and users, in order to predict a set of items that a user (or a group of users) will eventually like. But these patterns may also be utilized to obtain additional useful knowledge about the tastes of users in general or about the affinities of items.

For this purpose it is common to use clustering techniques. From the point of view of predictions, clusters are useful because they provide more reliable information than single users or items. The overall idea is that a group of items can be suggested to a group of users; this is not a very precise recommendation but it is hopefully in the right way. Typically, cluster-based recommenders promise a greater robustness and make scalable some techniques that would otherwise be inapplicable.

Moreover, knowing about their customers (users) is important for companies. This knowledge is usually aggregated in *market segments*, which allows companies to offer differentiated products and after-sales services. Furthermore, marketing messages can be more focused on the segments that maximize the profits. Let us remark that, frequently, this requires to cluster items too.

Sometimes, we may find a dataset where users are described by a vector of feature values, but this is not the general case. Moreover, the available information about users is typically too scarce and limited to a small set of demographic values that are not enough to explain the behavior of users. Therefore, we need to include some data into the users' description if we want to gather users with similar tastes. As it is going to be detailed in next sections, the ratings given by users to items are frequently included in users features. In this way, the correlation or cosine gives rise to the similarity of users' interaction with items. The flaw of this approach is that datasets are very sparse, that is, the set of items rated by two users is usually too small to generalize their similarity. In some cases, the intersection between the sets of products that users have rated may even be empty, making it impossible to apply these traditional methods.

We present a general framework to make clusters of users and to measure their quality according to their interactions with items. Let us emphasize that we can use this framework even in the case that users or items are not described by a vector of feature values. That is to say, from the point of view of items, our proposal is applicable for *content based* recommenders but also for collaborative (*agnostic content*) filtering settings. Analogously, we do not require users to be represented by features for our proposal to be applicable.

To illustrate this methodology, we used seven public domain datasets: six of them are commonly used as benchmark for recommendation tasks, together with a dataset of readers and promoted news in digital publications. The purpose in the last one is to find a model able to predict the preferences of readers to click on a given link instead of other alternatives. In this dataset, readers do not rate promoted news or have the possibility to access all of them: each reader is shown a reduced set of promoted news and can click at most one of them. Hence, to elicit the relationship between users and items we learn a utility function by solving a preference learning task [3, 17, 21, 27]

The paper is organized as follows. In the next section we cite some state-of-the-art clustering methods in the context of RS. Next, we present an entropy-based measure to assess the quality of clusters of users with respect to their preferences. Section 4 describes some alternatives to represent users in the context of recommender systems with the aim of improving the clustering process. Then, we describe a formal setting to implement a recommender system applicable to the typical binary classification task of prediction whether a user likes/dislikes an item. This setting is then adapted to make it applicable to recommendation problems where the preferences of users are not expressed by rating items, but by choosing one item among others, which lead us to use a preference learning approach.

Before concluding this Introduction, we would like to briefly comment on the most important novelties and contributions of this paper (which will be discussed in more detail in Section 2.1): i) we present different ways of encoding users according to their product preferences in the context of recommender systems even when the data set is very sparse (encodings that can then be used to perform user clustering), and ii) we show a way to measure the quality of user clusters based only on the preferences shown by the users. The purpose of this paper is not to present a new clustering algorithm, but rather to present two tools (preference-based user encodings and a cluster quality measure) that are useful for any clustering algorithm in the context of recommender systems.

2 Related work

User clustering in RS starts from a vectorial description of users on which different algorithms are applied. We can distinguish two major types of approaches. The first one requires users to be described by vectors containing intrinsic properties of the users, such as their gender, age, etc. Thus, this approach is independent of the behavior of users in relation to the items involved in the RS [8, 19].

However, a second approach raises because most authors are inclined to represent users by the ratings they give to the items with which they interact. To measure the similarity between users, they employ measures such as correlation [29] or cosine [4]. The problem with these approaches is that the reliability of the similarity (whichever measure is used) is limited between those users which have not rated the same items. Unfortunately, this situation occurs in most cases. Furthermore, in some cases the ratings may not be homogeneous, which makes the task even more difficult. In this regard, [20] address the sparsity problem of heterogeneous recommendation, presenting a method for transferring knowledge from a domain in which consumers rate products using like/dislike to another domain in which ratings are made using a 5-star system. This method, however, still does not solve the problem of reliability of the similarity.

Surveys about the use of clusters of users are [4, 6, 19]. Here we find many references discussing also the methods to cluster users once they are represented. These methods range from deterministic algorithms as k-means [12] or hierarchical methods [15] to soft methods as c-means [22, 40].

Sometimes, clustering users in an RS is made to gain insight into the kind of customers of a given company, as in [38]. The goal here is that companies can improve their customer-relations management by enhancing customer satisfaction and loyalty. It is also sometimes interesting to cluster items in order to have a better understanding of them, as in [34], where a recommender system for Indian classical music is presented. This system makes recommendations to users based on their listening history and also analyzes and groups melodies based on note structures available in Indian classical music.

Nevertheless, the aim of using clustering techniques is frequently to improve robustness or scalability in RS. This is the case of [28] that combines a clustering technique with an associative classification algorithm to personalize recommendations. The clustering of

Table 1 This table shows the most common user encoding strategies found in recommender systems literature

Encoding	Based on user prefs.?	Needs learning?	Useful under sparsity?	Lots of items?	Example references
Demographic features	×	×	✓	✓	[8, 28]
User-item ratings	✓	×	×	×	[4, 28, 29, 34, 36]
Embeddings	✓	✓	✓	✓	[5, 27, 32, 33, 38, 40]

The meaning of each column is the following: *Based on user prefs.?*: the encoding takes into account user preferences *Needs learning?*: there is a learning process needed to obtain the encoding. *Useful under sparsity?*: the encoding can be reliably used under high sparsity conditions of the user-item matrix. *Lots of items?*: the encoding can be used in problems with a large number of items. The last column contains some bibliographic references where these encodings were used

users is built upon their demographic information together with the interaction with the items. It is also the case of [36], which is capable of making better movie recommendations by obtaining clusters with k-means and using an adaptive genetic neural network. In this work users are encoded only by the ratings they assigned to the movies.

In [30, 31], the authors use a clustering to obtain groups of items in the so-called *long tail* with more ratings than the individual items. In this way, an RS will consider to suggest those items that otherwise will be kept unknown for users since they can not accumulate enough ratings so as to be more frequently recommended. Focusing on efficiency, a detailed study of the difference in efficiency between using user clusters and not using them is shown in [25]. That paper shows that using clusters makes systems much faster.

[5] and [40] introduce a co-clustering approach (*Multiclass Co-Clustering, MCoC*) to find meaningful subgroups. The idea is to capture the relations of user-to-item, user-to-user, and item-to-item simultaneously. For this purpose the authors define an embedding of users and items into a common low-dimensional Euclidean space. The goal is to obtain close representations of users and items if they have a high rating. These embeddings are usually obtained using matrix factorization techniques [24] or an intermediate layer in a deep neural network [41]. Another co-clustering method is presented by [39], *CCCF, Co-Clustering for Collaborative Filtering*. It also uses the user-item matrix to infer, using a Bayesian point of view, a cluster of users and, at the same time, of items. Similar approaches involving the computation of a common embedding to map users and items into the same space has also been used in other fields, like education [32, 33] or the food industry [27].

2.1 Differences with our approach and contributions

Table 1 shows a summary of the encoding approaches typically used to represent users. The first one consists of using demographic data such as age, sex, address, etc. to represent and cluster users. This approach presents two weakness points: i) this information is sometimes missing or unreliable and ii) these features are not necessarily related to the preferences of the users. Although this encoding approach can be used in very large and sparse (in terms of evaluations) datasets, the latter issue is crucial because it renders useless clusters, poorly related to the users' tastes.

Another frequently used encoding consists of representing users by the ratings given to the items they assess. This representation clearly takes into account the users' preferences.

However, it is unusable when the amount of items is very large because the user-item matrix tends to be very sparse, recording little intersection between items assessed by different users, thus, yielding nonsense clusters.

Finally, users can be represented by their projection in a vector space by means of an embedding learned from the preferences expressed by the users. This approach is very popular because of its ability to deal with very large datasets, despite their sparsity. However, users with few items rated, or whose ratings were mostly given to items rated by few users, may have little influence in the learning of the embedding, so their encoding can be rather inaccurate.

We propose to use an utility function to encode the users. This utility function, which is aimed at predicting an score given by a user to an item, will be obtained by means of a process which learns from the user preferences. The resulting encoding can be used in problems with a very large number of items and where the user-item matrix is extremely sparse. We show in section 6 that clustering approaches using user encodings based in our utility function outperform those using other encodings.

Let us recall that we do not present any new clustering technique in this paper, nor do we intend to present a new recommender system. To be precise, the main contributions of this work are:

1. Several utility-based encoding variants of the users in such a way that their preferences regarding the items can be taken into account by any existing clustering method. This encoding approach can be used independently of the characteristics of the problem at hand.
2. A measure to assess the quality of the clusters so obtained. This measure takes into account the similarity of the users' preferences grouped in the same cluster by means of the entropy.

3 The quality of clusters

The quality of a cluster is generally assessed by measures related to the similarity among their elements with respect to a certain property. Our main objective is to obtain clusters of users with similar tastes with respect to a collection of items, that is, we need to assess the homogeneity with respect to their preferences.

Let us consider a general setting of a problem typically approached by recommender systems, where the set of users is represented by $\mathcal{U}$, and the set of items by $\mathcal{I}$. In this context, an RS is a map from $\mathcal{U}$ to the set of non-empty subsets of $\mathcal{I}$ (RS: $\mathcal{U} \mapsto \mathcal{I}$). And we will represent a clustering of users carried out by any means as $\Omega(\mathcal{U}) = \{\mathcal{U}_1, \dots, \mathcal{U}_m\}$.

We are interested in assessing the quality of these m clusters with respect to the preferences of the users they contain. These preferences are registered in a user-item matrix which stores the opinions of users (typically in rows) with respect to the items (in columns). Users' opinions can be registered in the form of numerical scores, like/dislike, etc.

Therefore, a straightforward approach to assess the clusters' quality could be to evaluate the similarity of their opinions with respect to the items, as is usually done by collaborative filtering approaches. The problem is that, in general, this matrix is extremely sparse.

However, we can fill the gaps in the user-item matrix using a RS. The task of any RS is to infer a utility function,

$$\text{utility} : (\mathcal{U}, \mathcal{I}) \longrightarrow \mathbb{R}, \tag{1}$$

that hopefully estimates the score that a user $\boldsymbol{u}$ would give to an item $\boldsymbol{i}$ (utility$(\boldsymbol{u}, \boldsymbol{i})$). Once the matrix has been filled using the utility function we can measure the similarity of users according to their (estimated) preferences about items.

This procedure can be very costly because of the usually huge number of items. To avoid this inconvenient, we can look for a representative subset of items and estimate the rates on them (using the utility function), instead of using the whole set. The items chosen as representatives will act as *sensors*, so a sensor that receives a high score from the utility function will be indicating that the items around it are preferable to others that are close to a sensor that receives a lower rating. Therefore, it is important to make an appropriate selection of the sensors (representative items) in such a way that they cover all the space in which the items are distributed. There are two possibilities:

- when we do not know any feature of the items we can take a sample of n items from the training set and use them as sensors, or
- when we know the properties of the items we can cluster them into n clusters according to those properties and use the centroids of each cluster as representative items.

In both cases we finish with the choice of n representative items that we will use as sensors: $\{\boldsymbol{s}_1, \ldots, \boldsymbol{s}_n\}$.

3.1 The measure

To formally define a measure of the quality of a clustering of users, let us assume the simplest case: the values of items for users are "like" or "don't like", in symbols $\{1, 0\}$. Then, for each cluster of users $\mathcal{U}_i \in \Omega(\mathcal{U})$ we can compute the proportion of likes for each $\boldsymbol{s}_j$ of the sensors (centroids if we were able to cluster items or samples otherwise):

$$p = \Pr(1|\mathcal{U}_i, \boldsymbol{s}_j) = \frac{|\{\boldsymbol{u} \in \mathcal{U}_i : \text{utility}(\boldsymbol{u}, \boldsymbol{s}_j) = 1\}|}{|\mathcal{U}_i|}.$$

In an ideal situation, when all users in $\mathcal{U}_i$ have the same preferences, this proportion would be 0 or 1. We can assess the homogeneity of the preferences by computing the uncertainty by means of the entropy presented by [37]:

$$H(p) = -p \, \log_2(p) - (1-p) \, \log_2(1-p). \tag{2}$$

Not all clusters $\mathcal{U}_i$ will host the same number of users, so we may compute a weighted average with all the values for all the clusters to obtain a quality score for the whole clustering. Thus, we define a *quality loss* of the clustering of users, $Cl(\mathcal{U})$, by

$$\mathrm{Q}_{loss}(\Omega(\mathcal{U})) = \sum_{i,j} \frac{|\mathcal{U}_i||\mathcal{I}_j|}{|\mathcal{U}||\mathcal{I}|} H(\Pr(+1|\mathcal{U}_i, \boldsymbol{s}_j)), \tag{3}$$

where $|\mathcal{I}_j|$ is the number of items in the $\mathcal{I}_j$ cluster (where $\boldsymbol{s}_j$ is the centroid acting as a sensor, when items were clustered), or 1 when the sensors (representative items) were sampled from $\mathcal{I}$.

Obviously, the lower the value of Q_{loss}, the better the clustering is, given that we are looking for clusters with low entropy.

It is important to remark here that the set of items $\mathcal{I}$ can be drawn from a test set instead of a training set. Thus, quality loss (3) can be used as any other evaluation measure in Machine Learning tasks.

4 User encoding

Sometimes we have descriptive information about users such as sex, age, zip code, etc. This information (*raw* features) may have some relationship with their tastes, for example, the interests of a middle-aged person are not the same as those of a teenager. Thus, we could make a clustering of users based on their features. However, these clusters are not taking into account the individual preferences of each person since they work with aggregations. The clusters obtained with the *raw* descriptions of users, $\boldsymbol{u}^{(\mathrm{raw})}$, will be denoted by $\Omega(\mathcal{U}^{(\mathrm{raw})})$.

In the previous section we have defined a reasonable measure to assess the quality of clusters, and inspired on it we may consider alternatives to $\boldsymbol{u}^{(\mathrm{raw})}$ to represent users. For instance, we may represent them using a vector with the scores estimated by the utility function for each sensor. That is,

$$\boldsymbol{u}^{(\mathrm{uti})} \leftarrow \left[\mathrm{utility}(\boldsymbol{u}, \boldsymbol{s}_j) : j = 1, \ldots, n\right]. \tag{4}$$

Thus, users will be represented by their (estimated) taste with respect to a subset of items (sensors). The clustering of users so obtained will be denoted by $\Omega(\mathcal{U}^{(\mathrm{uti})})$.

These two approaches, *raw* and *utility-based*, are the building blocks of other options. For instance, we may combine both representations. In symbols, we may represent users by

$$\boldsymbol{u}^{(\mathrm{raw}\oplus\mathrm{uti})} \leftarrow \boldsymbol{u}^{(\mathrm{raw})} \oplus \boldsymbol{u}^{(\mathrm{uti})}, \tag{5}$$

where $\oplus$ denotes the *direct sum* or concatenation of two vectors. This gives rise to a clustering that we will denote as $\Omega(\mathcal{U}^{(\mathrm{raw}\oplus\mathrm{uti})})$.

Finally, the utility-based representation of users (4) can be weighted by the number of items represented by each sensor (if the sensors are centroids of clusters of items, provided we have previously obtained such clusters). Thus, we can compute a vector of weights,

$$\boldsymbol{w} \leftarrow \left[\sqrt{\frac{|\mathcal{I}_j|}{|\mathcal{I}|}} : j = 1, \ldots, n\right] \tag{6}$$

and then we can use it to create a weighted version of their utility-based representation,

$$\boldsymbol{u}^{(\mathrm{w}\odot\mathrm{uti})} \leftarrow \boldsymbol{w} \odot \boldsymbol{u}^{(\mathrm{uti})}, \tag{7}$$

where $\odot$ is the element-wise product, also known as the Hadamard product. We denote by $\Omega(\mathcal{U}^{(\mathrm{w}\odot\mathrm{uti})})$ the clustering so obtained.

Note that, if the sensors (representative items) were randomly drawn from the set of items, the value of $|\mathcal{I}_j|$ will be 1. This causes all the clusters to have the same weights ($\sqrt{\frac{1}{|\mathcal{I}|}}$) and the results obtained by the user encodings (4) and (7) are equivalent. Thus, the encoding (7) only makes sense when the properties of the items are available and a proper clustering can be carried out, in order to select the centroids as sensors and to compute a sensible value for the weights, $\boldsymbol{w}$.

Other user encoding approaches have been frequently used in the literature. The most basic encoding can be obtained directly from the user-item matrix containing the ratings given by the users to the items. Using this approach, users will be represented by the row vectors of the user-item matrix. This encoding is typically used in user-based collaborative filtering tasks. When the number of items is large, this encoding can be inapplicable, not only due to the size of the vectors, but because the sparsity of the matrix becomes very large, making this encoding useless due to the lack of intersections in the assessment of items. We will refer to users encoded in this way as $\boldsymbol{u}^{(\mathrm{rate})}$, and the resulting clustering as $\Omega(\mathcal{U}^{(\mathrm{rate})})$.

Sometimes we may be interested in knowing if a user has *interacted* with an item, even thought the item was not rated at all. For example, an online store can be interested in the items that a user has been viewing, in order to provide a recommendation of related products (not necessarily similar but related in some way). For this purpose, the visits to the items must be recorded and then used, without even known the ratings that the user would eventually give to the items. These visits can be stored in a binary user-item matrix where we will have 1's to indicate interactions between users and items, and 0's to denote the lack of interaction. Using the row vectors of this binary matrix we have another encoding, $\boldsymbol{u}^{(\mathrm{int})}$, which will lead us to another clustering, $\Omega(\mathcal{U}^{(\mathrm{int})})$.

There is another encoding that has been used recently in the literature, which consists in computing an embedding to project the original representation of users in a new vectorial space. These projections are obtained attending to the preferences of users over the items. We will denote this encoding as $\boldsymbol{u}^{(\mathrm{emb})}$, and the resulting clusters obtained in this embedding will be denoted as $\Omega(\mathcal{U}^{(\mathrm{emb})})$. The embedding can be obtained using deep learning [2, 7, 9, 26, 41] or matrix factorization techniques [27, 32, 33]. In this paper, since we use matrix factorization to obtain the utility function (as we will see in the next section), we have decided to use this same technique to calculate the embedding that projects users in this new vectorial space. We have made this decision since deep learning based models usually require learning a larger number of parameters as well as fitting a larger number of hyperparameters, as opposed to matrix factorization which generates simpler models. We are aware that by means of a model based on deep learning we could obtain a slightly better utility function than the one obtained using matrix factorization, however, we have decided to sacrifice this improvement in the results that our proposal could have for the sake of a greater simplicity of the method presented in the paper.

In the experiments reported at the end of the paper, we show the quality loss (3) of all these clustering approaches in six public-domain recommender system benchmarks, as well as in the Outbrain dataset mentioned in the Introduction.

To conclude this section, let us remark that our proposed encodings $\boldsymbol{u}^{(\mathrm{uti})}$ and $\boldsymbol{u}^{(\mathrm{w}\odot\mathrm{uti})}$ may not use any feature values of users: the clusters $\Omega(\mathcal{U}^{(\mathrm{uti})})$ and $\Omega(\mathcal{U}^{(\mathrm{w}\odot\mathrm{uti})})$ can be computed from a simple one-hot representation of the users if we do not have access to the users' features. Therefore, the method we are proposing in this paper is able to encode users without knowing any user/item feature, allowing us to obtain reliable clusters afterwards.

5 Formal setting for a recommender system

In order to evaluate and compare the different representations suggested previously, we implemented a matrix factorization based Recommender System to infer a utility function able to predict the preferences of users. Let us recall that the utility function will allow us to build several user representations and to evaluate the quality of the clusters depending on that representations.

Firstly, we present the formal framework for a typical recommender system which learns to predict whether a given user would like a given item. Thus, the recommendation problem is posed as a classification task. Then, we extend the framework to deal with the recommendation task from a preference learning perspective, in which we will learn from the elections made by the users when they have to choose one item among several available. In both cases we use a matrix factorization approach but any other technique can be used, e.g., a deep learning approach, provided it yields an utility function (1).

Let $\mathcal{D}$ be a dataset containing the interactions of users with items. We assume that both users and items have a vectorial representation. The elements of $\mathcal{D}$ are tuples of 3 elements

$$\mathcal{D} = \{(\boldsymbol{u}, \boldsymbol{i}, \{1, 0\}) : \boldsymbol{u} \text{ evaluates } \boldsymbol{i} \text{ and likes (1) or dislikes (0)}\}, \tag{8}$$

where $\boldsymbol{u} \in \mathcal{U}$ is a vector representing the user and $\boldsymbol{i} \in \mathcal{I}$ is a vector representing an item evaluated by $\boldsymbol{u}$. These vectors can be built from the features of the users and items (in the case of content-based recommendation systems) or they can be represented by simple one-hot vectors, in case those features are unknown. For further references, $\mathcal{U}$ will be the set of users and $\mathcal{I}$ will be the set of items.

Following a matrix factorization approach, a first attempt to define the utility of an item $\boldsymbol{i}$ for a user $\boldsymbol{u}$ can be given by a function

$$\text{utility}_0(\boldsymbol{u}, \boldsymbol{i}) = \langle \boldsymbol{W}\boldsymbol{u}, \boldsymbol{V}\boldsymbol{i} \rangle. \tag{9}$$

Here, $\boldsymbol{W}$ and $\boldsymbol{V}$ are matrices that must fulfill the constraints gathered in $\mathcal{D}$, and the rating given by the utility function should be positive when the user likes the item and negative otherwise. Therefore, to learn the utility function, we use a logistic regression approach. Then, we consider the utility given by

$$\begin{aligned} \text{utility}(\boldsymbol{u}, \boldsymbol{i}) &= \Pr(1|\boldsymbol{u}, \boldsymbol{i}) \\ &= \sigma(\text{utility}_0(\boldsymbol{u}, \boldsymbol{i})) \\ &= \sigma(\langle \boldsymbol{W}\boldsymbol{u}, \boldsymbol{V}\boldsymbol{i} \rangle), \end{aligned} \tag{10}$$

where σ is the sigmoid function,

$$\sigma(x) = \frac{1}{1 + e^{-x}}.$$

Therefore, using a maximum likelihood approach, we need to solve the following optimization problem to find the matrices $\boldsymbol{W}$ and $\boldsymbol{V}$:

$$\underset{\mathbf{W},\mathbf{V}}{\text{argmin}} \sum_{\mathcal{D}} \text{softplus}(-\langle \mathbf{Wu}, \mathbf{Vi} \rangle) + \frac{\nu}{2}\left(||\mathbf{W}||^2 + ||\mathbf{V}||^2\right), \tag{11}$$

where $\text{softplus}(x) = \ln(1 + e^x)$, and ν is a regularization factor used to set up a trade-off between the classification errors and the complexity (in a broad sense) of the parameters that we are learning, i.e., $\boldsymbol{W}$ and $\boldsymbol{V}$.

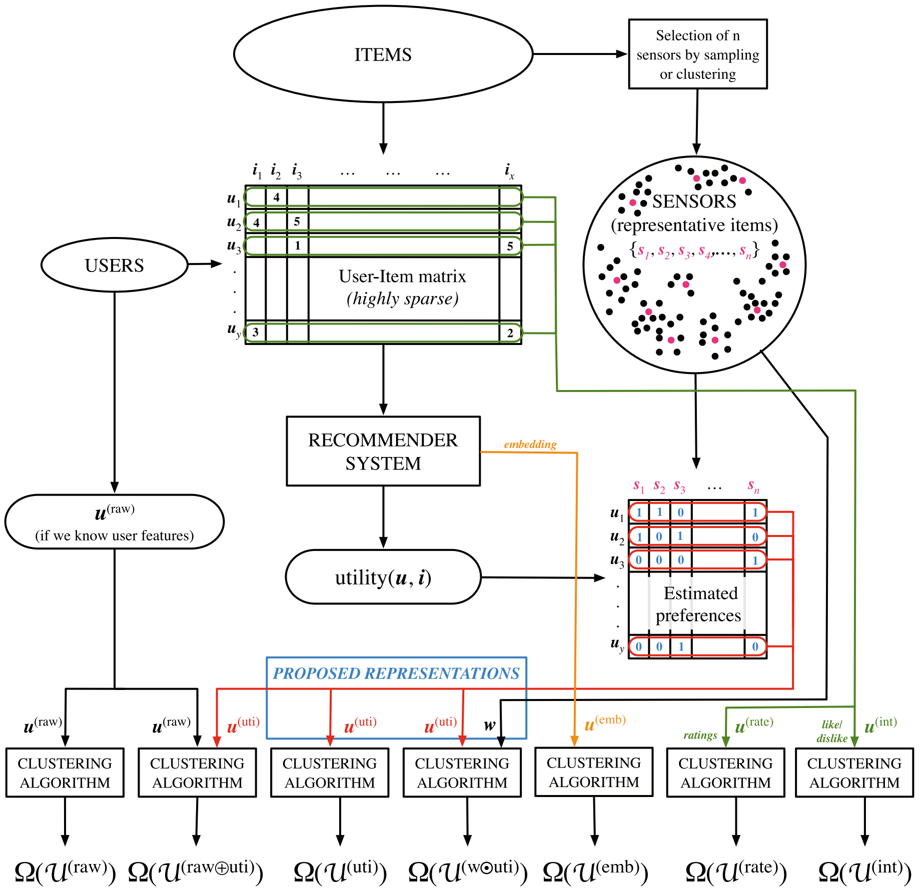

ITEMS
Selection of n sensors by sampling or clustering
SENSORS
(representative items)
$\{s_1, s_2, s_3, s_4, \ldots, s_n\}$
USERS
i_1 i_2 i_3 … … … i_x
u_1
u_2
u_3
u_y
4
4 5
1 5
3 2
User-Item matrix
(highly sparse)
RECOMMENDER SYSTEM
embedding
utility($\boldsymbol{u}, \boldsymbol{i}$)
$\boldsymbol{u}^{(\mathrm{raw})}$
(if we know user features)
s_1 s_2 s_3 … s_n
1 1 0 1
1 0 1 0
0 0 0 1
0 0 1 0
Estimated preferences
PROPOSED REPRESENTATIONS
$\boldsymbol{u}^{(\mathrm{raw})}$
$\boldsymbol{u}^{(\mathrm{raw})}$
$\boldsymbol{u}^{(\mathrm{uti})}$
$\boldsymbol{u}^{(\mathrm{uti})}$
$\boldsymbol{u}^{(\mathrm{uti})}$
$\boldsymbol{w}$
$\boldsymbol{u}^{(\mathrm{emb})}$
ratings
$\boldsymbol{u}^{(\mathrm{rate})}$
like/ dislike
$\boldsymbol{u}^{(\mathrm{int})}$
CLUSTERING ALGORITHM
$\Omega(\mathcal{U}^{(\mathrm{raw})})$
$\Omega(\mathcal{U}^{(\mathrm{raw}\oplus\mathrm{uti})})$
$\Omega(\mathcal{U}^{(\mathrm{uti})})$
$\Omega(\mathcal{U}^{(\mathrm{w}\odot\mathrm{uti})})$
$\Omega(\mathcal{U}^{(\mathrm{emb})})$
$\Omega(\mathcal{U}^{(\mathrm{rate})})$
$\Omega(\mathcal{U}^{(\mathrm{int})})$

Fig. 1 Overall approach: we infer the estimated preferences of the users on a subset of representative items (sensors) using the utility function learned by the recommender system. The filled matrix gathers the (inferred) behavior of users, which can be used to encode the users and favor the clustering regarding their preferences. The graph shows how our proposed encodings are obtained, as well as other commonly used representations

Once we have learned the utility function (10) we can use it to estimate the opinion of any user with respect to any item. In particular, we are going to use it to infer the values that each user would give to each of the representative items (sensors) obtained from the set of items $\mathcal{I}$. These values will then be used to build alternative representations of users, aimed at obtaining more homogeneous clusters in terms of users' preferences, as stated in Section 4. Figure 1 shows the overall approach, which we have used to compare the performance of the different user encodings in several public domain datasets commonly used as benchmarks in recommendation tasks. The results of the comparison are discussed in Section 6.1.3.

5.1 Formal setting for a preference-based recommender system

We can apply the framework presented previously to recommendation tasks based on preference learning. In this case we do not know the ratings of items given by users. Instead, we only know that the user chose an item among a (usually small) set of items, but we do not know/have the rating, neither of the chosen item nor of the others.

In order to test our proposal in a preference learning scenario, we faced a problem posed by the company Outbrain through the Kaggle platform, which consists of predicting the promoted link that a reader will choose after reading a piece of news in a web site. In the following we adapt our previously presented framework to the specific characteristics of this task.

Let $\mathcal{D}$ be, in this case, a dataset that records the interactions of readers with promoted links that are suggested when reading a digital publication. We assume that both readers and documents have a vectorial representation. The elements of $\mathcal{D}$ are tuples of 4 vectors

$$\mathcal{D} = \left\{ (\boldsymbol{r}, \boldsymbol{d}, \boldsymbol{l}_1, \boldsymbol{l}_2) : \begin{array}{c} \boldsymbol{r} \text{ is reading } \boldsymbol{d} \text{ and} \\ \text{clicks in } \boldsymbol{l}_1 \text{ instead of } \boldsymbol{l}_2 \end{array} \right\}, \tag{12}$$

where $\boldsymbol{r}$ is the representation of the reader, $\boldsymbol{d}$ is the document that $\boldsymbol{r}$ reads while a set of promoted links are suggested (the reader is only presented with between 4 and 6 promoted links and can click at most one of them), $\boldsymbol{l}_1$ is the content of the link clicked by $\boldsymbol{r}$, and $\boldsymbol{l}_2$ is the content of another link that was not clicked.

In order to predict the link that a user is going to click, we should consider the reading context. Thus, each user will be represented by the concatenation of its own representation (based on features or a simple one-hot) together with the vector representing the document the user is reading, $\boldsymbol{u} = \boldsymbol{r} \oplus \boldsymbol{d}$, when they click on a promoted link. Thus, we include the reading context as part of the user profile. For further references,

$$\mathcal{U} = \{(\boldsymbol{r} \oplus \boldsymbol{d}) : \boldsymbol{r} \text{ is a reader}, \boldsymbol{d} \text{ is a document}\}. \tag{13}$$

The utility function (10), must be redefined as the utility of a link $\boldsymbol{l}$ for a reader $\boldsymbol{r}$ who is reading the news item $\boldsymbol{d}$,

$$\text{utility}_0(\boldsymbol{u}, \boldsymbol{i}) = \text{utility}_0(\boldsymbol{r} \oplus \boldsymbol{d}, \boldsymbol{l}) = \langle \boldsymbol{W}(\boldsymbol{r} \oplus \boldsymbol{d}), \boldsymbol{V}\boldsymbol{l} \rangle, \tag{14}$$

where $\boldsymbol{u} \in \mathcal{U}$ and $\boldsymbol{i} \in \mathcal{I}$. As indicated in Section 5, $\boldsymbol{W}$ and $\boldsymbol{V}$ are matrices which must fulfill the constraints gathered in $\mathcal{D}$; in this case, the utility of the chosen link, $\boldsymbol{l}_1$, must be higher than that of $\boldsymbol{l}_2$,

$$\langle \boldsymbol{W}(\boldsymbol{r} \oplus \boldsymbol{d}), \boldsymbol{V}\boldsymbol{l}_1 \rangle > \langle \boldsymbol{W}(\boldsymbol{r} \oplus \boldsymbol{d}), \boldsymbol{V}\boldsymbol{l}_2 \rangle, \quad \forall (\boldsymbol{r}, \boldsymbol{d}, \boldsymbol{l}_1, \boldsymbol{l}_2) \in \mathcal{D}.$$

Hence,

$$\langle \boldsymbol{W}(\boldsymbol{r} \oplus \boldsymbol{d}), \boldsymbol{V}(\boldsymbol{l}_1 - \boldsymbol{l}_2) \rangle > 0. \tag{15}$$

This approach for *learning preferences*, which was used in [3, 17, 21, 27], reduces the learning task to a binary classification.

The consequences of inequality (14) are very useful. From the point of view of RS, we have a dataset where items are differences of the vectorial representation of the promoted links. Thus, we can redefine the item set as

$$\mathcal{I} = \{(\boldsymbol{l}_1 - \boldsymbol{l}_2) : \boldsymbol{l} \text{ is the representation of a link}\}.$$

Finally, the rating given by a user to such items (Eq. 15) is positive if the reader prefers the first link to the second one, and negative in the other case. Therefore, we adapt the definition of the utility function (10) as follows:

$$\begin{aligned}\text{utility}(\boldsymbol{u},\boldsymbol{i}) &= \text{utility}(\boldsymbol{r}\oplus\boldsymbol{d},\boldsymbol{l}_1-\boldsymbol{l}_2)\\ &= \Pr(1|\boldsymbol{r},\boldsymbol{d},\boldsymbol{l}_1-\boldsymbol{l}_2)\\ &= \sigma(\text{utility}_0(\boldsymbol{r}\oplus\boldsymbol{d},\boldsymbol{l}_1-\boldsymbol{l}_2))\\ &= \sigma\big(\langle \boldsymbol{W}(\boldsymbol{r}\oplus\boldsymbol{d}),\boldsymbol{V}(\boldsymbol{l}_1-\boldsymbol{l}_2)\rangle\big).\end{aligned} \tag{16}$$

Note that given the symmetry of the sigmoid, σ, and the constraint (15), we only need to consider positive examples. Then, the optimization problem to find the matrices $\boldsymbol{W}$ and $\boldsymbol{V}$ is as follows:

$$\underset{\mathbf{W},\mathbf{V}}{\text{argmin}}\sum_{\mathcal{D}}\text{softplus}\big(-\langle\mathbf{W}(\mathbf{r}\oplus\mathbf{d}),\mathbf{V}(\mathbf{l}_1-\mathbf{l}_2)\rangle\big)+\frac{\nu}{2}\big(\|\mathbf{W}\|^2+\|\mathbf{V}\|^2\big). \tag{17}$$

The results obtained using different user encodings in this problem are presented and discussed in Section 6.2.2.

6 Experimental results

This section is devoted to present the experimental results obtained in clustering tasks, in the context of recommender systems, in order to compare the performance of the different user encodings presented in Section 4. The section is organized in two parts: the first one shows the results obtained on a group of public-domain datasets, commonly used as benchmark for recommender systems. The second one shows the results obtained on the Outbrain dataset which poses a different learning task, as explained in Section 5.

6.1 Results obtained in the benchmark

In the following we present the most relevant characteristics of the datasets used as benchmark. We also reveal some implementation details and, finally, we present and discuss the results obtained in the experimentation.

6.1.1 Description of the datasets

We have collected the most remarkable characteristics of the datasets used in this benchmark in Table 2. Notice the variety in the number of users and items. Also, the number of evaluations made by users is scarce in all but one dataset, which results in very sparse user-item matrices. On the other hand, our approach is able to deal with problems whose user/item features are unavailable, so we have also included some datasets with this lack of information, in order to evaluate the performance of the different encoding approaches.

- **MovieLens**[1] [16]. These datasets collect the ratings given to movies, with the restriction that each user rated at least 20 movies and each movie was rated by at least 20

[1] https://grouplens.org/datasets/movielens/

Table 2 Main characteristics of the public-domain datasets used in this benchmark

Data set	#users	#items	#ratings	density	user feat.	item feat.	likes
MovieLens100K	943	1,682	100,000	6.30%	✓	✓	56.8%
MovieLens1M	6,040	3,838	1,000,209	4.26%	✓	✓	55.8%
Book-Crossing	6,029	5,633	92,449	0.27%	✓	✓	63.0%
CiaoDVD	17,615	16,121	72,665	0.03%	×	✓	44.7%
Jester	73,421	100	4,136,360	56.34%	×	×	58.4%
Filmtrust	1,508	2,071	35,497	1.14%	×	×	45.6%

users. We have descriptive information of the users (age, gender, occupation, zipcode) and the items (one or more genre(s) of the movie, among 18 possible) in the two datasets used. The zipcode was transformed to GPS coordinates (latitude and longitude). The ratings range from 1 to 5.

- **Book-Crossing**[2] [42]. This dataset is made of the ratings of users to books. The users are described by their age and location, and the books by author, publisher and year of publication. In this case, the location is given by the name of the city, which was recorded very imprecisely and not curated at all, thus making it useless. Moreover, the age of aproximately one third of the users is unknown, so the mode was imputed. Most of the users and books in the dataset participated only in one evaluation so we filter the dataset, originally containing more than 270,000 books and users, leaving only those users with at least 5 evaluations, and books with at least 10 evaluations. The resulting size is reflected in the table. The ratings of the books range between 1 and 10.
- **CiaoDVD**[3] [13]. This dataset is also about rating movies, in this case in DVD format. The dataset contains no information about the users and only the genre(s) of the movies. We filtered out the users and movies with no ratings, resulting in the figures shown in the table. The ratings range from 1 to 5.
- **Jester**[4] [11]. This dataset contains ratings of jokes. There is no information at all about users and items (jokes), only their identification number. The ratings range from -10 to 10.
- **Filmtrust**[5] [14]. Another movie rating dataset, in which there is no information about users and items. The ratings vary from 0.5 to 4.

6.1.2 Implementation details

In order to create the dataset $\mathcal{D}$ (8) for each problem, we built the triplets $(\boldsymbol{u}, \boldsymbol{i}, x)$, where x is either 1 (like) or 0 (dislike) depending on the rating given by user $\boldsymbol{u}$ to item $\boldsymbol{i}$. Following [18], we considered ratings below 4 as dislike and ranking greater or equal to 4 as like for the MovieLens and CiaoDVD datasets. For datasets with different ratings range we tried to get a balance between likes and dislikes. Thus, in Book-Crossing all ratings below 7,

[2] http://www2.informatik.uni-freiburg.de/~cziegler/BX/

[3] https://guoguibing.github.io/librec/datasets/CiaoDVD.zip

[4] https://goldberg.berkeley.edu/jester-data/

[5] https://guoguibing.github.io/librec/datasets/filmtrust.zip

in Jester all negative values and in Filmtrust all ratings below 3.5 are considered dislike. Table 2 shows the final class balance for each dataset.

The vectors representing the users, $\boldsymbol{u}$, and the items, $\boldsymbol{i}$, are built from the information available in each case. The features latitude, longitude, age and year were standardized where available. For those datasets with no information other than the identification of users and items we used a simple one-hot codification.

We split the dataset $\mathcal{D}$ into train and test, reserving 25% of the items together with their ratings for testing. The utility function (10) was learned using a TensorFlow [1] implementation with the following hyperparameters:

- The dimension of the embedding where users and items are projected was 256, i.e., the number of rows of matrices $\boldsymbol{W}$ and $\boldsymbol{V}$ (see Section 5),
- The regularization factor (11) was $\nu = 10^{-5}$,
- The batch size was set to 512,
- The learning rate was 10^{-4} and,
- To solve the optimization problem (11) we used Adam [23].

Regarding the clustering, we applied the K-means implementation of [35] (*Mini-batch K-means*) with a batch size of 1024 to cluster the users using all the encoding options described in Section 4. The results reported were obtained with 10 clusters.

To obtain the sensors (representative items), we performed a clustering in the problems which included some sort of description of the items. We used the same algorithms and hyperparameters as those used to cluster users. In the rest of problems, where clustering makes no sense due to the lack of items' features, we randomly selected 10 items. Thus, in these experiments, all published results were obtained using 10 sensors ($n = 10$).

6.1.3 Results and discussion

In Table 3, we report the results achieved in terms of Q_{loss}, the quality loss (3), for each of the user encodings presented in Section 4. Let us recall that we are measuring the average uncertainty, so the lower is the better.

Let us recall that $\Omega(\mathcal{U}^{(\text{raw})})$ and $\Omega(\mathcal{U}^{(\text{raw}\oplus\text{uti})})$ cannot be computed for those datasets with unavailable user information. When the information of the items is missing, $\Omega(\mathcal{U}^{(\text{w}\odot\text{uti})})$ cannot be computed either.

Table 3 Quality loss (3) achieved by the different user encodings for each benchmark dataset (the lower, the better). The name of each column indicates the encoding approach used

Data set	$\Omega(\mathcal{U}^{(\text{raw})})$	$\Omega(\mathcal{U}^{(\text{raw}\oplus\text{uti})})$	$\Omega(\mathcal{U}^{(\text{uti})})$	$\Omega(\mathcal{U}^{(\text{w}\odot\text{uti})})$	$\Omega(\mathcal{U}^{(\text{emb})})$	$\Omega(\mathcal{U}^{(\text{rate})})$	$\Omega(\mathcal{U}^{(\text{int})})$
MovieLens100K	0.3995	0.3259	0.2608	**0.2574**	0.3784	0.4632	0.4618
MovieLens1M	0.1846	0.1846	0.1128	**0.1045**	0.1861	0.2159	0.2157
Book-Crossing	0.7436	0.6953	0.2690	**0.2044**	0.7333	0.7540	0.7551
CiaoDVD	–	–	0.7473	**0.7382**	0.7737	0.9962	0.9963
Jester	–	–	**0.2822**	–	0.2933	0.6077	0.8870
Filmtrust	–	–	0.8896	–	**0.8875**	0.9745	0.9766

The best clusters, in terms of Q_{loss}, are $\Omega(\mathcal{U}^{(\text{w}\odot\text{uti})})$ for all datasets where $\boldsymbol{u}^{(\text{w}\odot\text{uti})}$ can be computed. The second best clustering in these datasets is $\Omega(\mathcal{U}^{(\text{uti})})$. Moreover, it is the best in one of the two datasets with no user information (Jester), and obtains a very similar result with respect to $\Omega(\mathcal{U}^{(\text{emb})})$, which is the best in the other dataset (Filmtrust).

On the contrary, the worst results are obtained when using the user encodings $\boldsymbol{u}^{(\text{rate})}$ and $\boldsymbol{u}^{(\text{int})}$, which makes sense because these encodings register almost no intersection (relation) between users due to the high sparsity of the user-item matrices. The only dataset in which $\Omega(\mathcal{U}^{(\text{rate})})$ is clearly better than $\Omega(\mathcal{U}^{(\text{int})})$ is Jester, which has a much denser user-item matrix. This also makes sense, given that a rating-based encoding records a richer information regarding the tastes of the users than encoding using only a binary value to indicate the interaction of users with items.

Using the $\boldsymbol{u}^{(\text{raw})}$ encoding is not a good option either. The results with the clustering $\Omega(\mathcal{U}^{(\text{raw})})$ are not the worst, but they are far from the best. Nevertheless, we observe that this encoding yields better results in the MovieLens datasets than in the Book-Crossing dataset. This is due to the fact that the users' description in the latter is poorer than in the former (neither gender nor occupation was registered, and the age is missing for many users). Notice also that incorporating the utility in this encoding, i.e. using $\boldsymbol{u}^{(\text{raw}\oplus\text{uti})}$, improves the results in two out of three datasets.

In view of the results of this benchmark, we recommend the $\boldsymbol{u}^{(\text{w}\odot\text{uti})}$ encoding as long as items' descriptions are provided (they are needed to weight the utility function). Otherwise, we propose to use the $\boldsymbol{u}^{(\text{uti})}$ encoding.

Finally, we want to highlight that the improvement in the clustering performance achieved with the $\boldsymbol{u}^{(\text{w}\odot\text{uti})}$ encoding is higher when the user-item matrix is very sparse.

6.2 Results in a preference learning context

In the following we present the performance achieved by several user encodings in clustering tasks on a dataset which has a couple of remarkable differences with respect to those used as benchmark in the previous section:

- Users did not rate any item; instead, they chose the most appealing one among a small set of items, and
- The huge amount of items together with the extremely low density of the user-item matrix makes the $\boldsymbol{u}^{(\text{rate})}$ and $\boldsymbol{u}^{(\text{int})}$ encodings unusable.

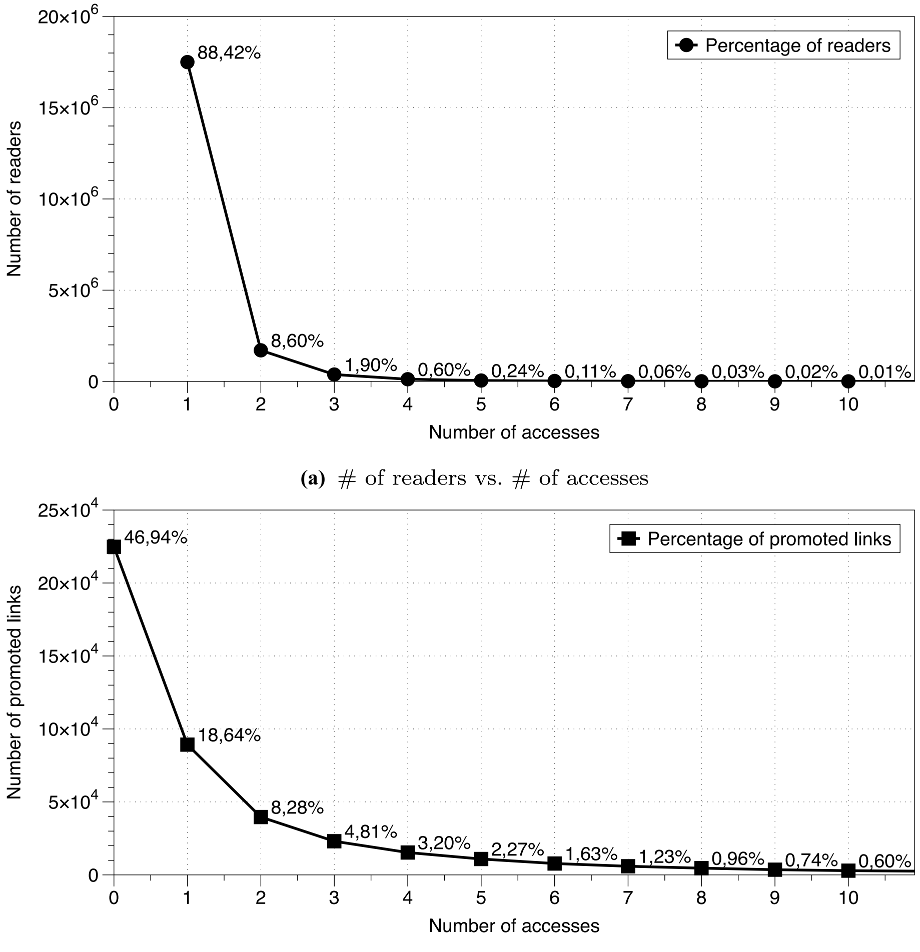
Firstly, we describe the main characteristics and the preprocessing of the dataset used in the experiments. Then, in Section 6.2.2, we report the scores achieved by the clustering algorithm for several user encodings. These results have been compared using the Bergmann's test throughout the data, which has been split in datasets of 24 hours of readers' interactions with digital publications.

6.2.1 Description of the outbrain dataset

In the experiments reported in this section we used a dataset deployed by the company Outbrain[6] for a Kaggle challenge[7]. It is a sample of users' page views and clicks on multiple

[6] https://www.outbrain.com

[7] https://www.kaggle.com/c/outbrain-click-prediction/data



(a) # of readers vs. # of accesses

(b) # of promoted links vs. # of accesses

Fig. 2 These graphs illustrate the enormous sparsity in the data. Figure 2a depicts the number of readers vs. the number of accesses to the promoted links; for example, 17.5 million readers (88.4% of the total) clicked only one promoted link. Figure 2b shows the number of promoted links vs. the number of times they were accessed; for example, 224,814 links (46.9% of the total) do not have any access. To facilitate the reading of the graph, we only show data with less than 10 accesses. In the case of readers, only 0.024% have made more than 10 accesses to promoted links and, in the case of links, only 10.7% of the total promoted links received more than 10 accesses

publisher sites around the world in June 2016, between the 14th and the 28th. The bottom of the pages read by the users include a group of links to promote other readings (usually 5 links). Most of the times, the readers do not click on any link, but sometimes they access one of them. Each viewed page or clicked recommendation is further accompanied by some semantic attributes.

In this article we will work only with those groups of promoted links in which some reader has clicked at least on one link. Thus, we will have a total of 559,584 promoted links and 19,794,967 readers who have accessed any of those links. In total, 23,120,126 accesses were registered, which means that only 0.00021% of the co-occurrence matrix between readers and promoted links is filled. It is therefore a recommendation task where the user-item matrix is extremely sparse compared to other recommendation tasks commonly used in the literature (see Table 2). For this reason, traditional collaborative filtering encodings ($\boldsymbol{u}^{(\text{rate})}$ and $\boldsymbol{u}^{(\text{int})}$) cannot be applied in this problem.

We can see in Fig. 2a that 17.5 million readers (88.4% of the total) accessed only one promoted link. The percentage drops dramatically to 8.6% for users who have accessed 2 links. Figure 2b shows that 224,814 promoted links (46.9% of the total) do not have any access and 18.6% of them were accessed only once.

These numbers are useful to highlight the difficulty of the task we deal with. Traditional collaborative filtering approaches, which typically use cosine or correlation-based similarity measures, are not applicable in this problem due to its high sparsity: the vast majority of pairs of users will have no common links, thus yielding 0 similarity score. This information is completely useless in order to cluster users with respect to their preferences.

We used readers, documents and promoted links as defined in Section 5.1. Their representations are composed as a concatenation of the representation of some fields listed in Table 4. Latitude and longitude are represented by a single real number corresponding to the GPS coordinates of the center of the State or Country. We normalize these values in order to keep them between 0 and 1. The other fields are binary vectors that codify the presence or not of the features that are present in at least 2500 documents. These vectors may have several ones, for instance when a document has several topics, entities or categories. After the binary representation, we multiply them by the confidence that Outbrain assigns to those values (a value in the range [0, 1]). Table 4 shows the dimensions of all the fields used in the experiments.

As we mentioned earlier, each group of promoted links is usually made up of 5 links, where the reader accesses the link that most attracted his/her attention and does not access the rest. Thus, we can generate several elements for the set $\mathcal{D}$ (12) indicating that the link accessed is preferable for the reader over the rest of links in that group. After analyzing all the accesses recorded, the set $\mathcal{D}$ was formed by 70,267,138 elements.

For the experiments we divided the dataset in blocks of 24 hours. So, we obtained 13 datasets that were finally split trying to have between 90% and 95% for training and the rest

Table 4 Documents include information about the publisher, category, entity and topic, as well as Outbrain's confidence in each respective relationship. Each field is represented as a binary vector multiplied by the corresponding confidence. We only used features that appear in at least 2500 items

Field	*Dimension*	*Comment*
document	662	
publisher	190	weighted by the confidence
category	83	weighted by the confidence
entity	99	weighted by the confidence
topic	290	weighted by the confidence
reader	5	
latitude	1	normalized in [0, 1]
longitude	1	normalized in [0, 1]
platform	3	3 possible platforms
link	699	
advertiser	37	
document	190+83+ 99+290 =662	weighted by the confidence

Table 5 Sizes of datasets (number of examples) used in the experiments reported in this section

Day	train	test
1	5,500,000	538,092
2	5,000,000	371,410
3	5,250,000	325,203
4	5,000,000	244,993
5	4,250,000	314,621
6	4,250,000	337,243
7	5,500,000	532,854
8	5,250,000	434,170
9	5,250,000	518,113
10	5,250,000	497,922
11	5,250,000	379,320
12	4,500,000	336,755
13	4,750,000	436,442

for testing. According to these specifications, the sizes of the resulting datasets are shown in Table 5.

Regarding the user clustering task, we set the corresponding hyperparameters as indicated in Section 6.1.2, with the exception of the number of target clusters, which in this case was 100 due to the large amount of users and items (promoted links) in the datasets (we also checked with different numbers of clusters, obtaining similar results). To obtain the representative items we also performed a clustering because they were described by

Table 6 Quality loss (3) achieved with each dataset (each day) by the different user encodings (the lower, the better). The name of each column indicates the representation used for users. The bottom line shows the average ranking position in the performance ranking of the five encodings. In brackets the position in the ranking of each encoding for each day

Day	$\Omega(\mathcal{U}^{(\mathrm{raw})})$	$\Omega(\mathcal{U}^{(\mathrm{raw}\oplus\mathrm{uti})})$	$\Omega(\mathcal{U}^{(\mathrm{uti})})$	$\Omega(\mathcal{U}^{(\mathrm{w}\odot\mathrm{uti})})$	$\Omega(\mathcal{U}^{(\mathrm{emb})})$
1	0.2978 (5)	0.2579 (2)	0.2582 (3)	0.2543 (1)	0.2814 (4)
2	0.2924 (5)	0.2445 (3)	0.2433 (2)	0.2388 (1)	0.2740 (4)
3	0.3651 (5)	0.3029 (2)	0.3040 (3)	0.3012 (1)	0.3369 (4)
4	0.2558 (5)	0.2185 (3)	0.2182 (2)	0.2166 (1)	0.2381 (4)
5	0.2487 (5)	0.2173 (2)	0.2188 (3)	0.2123 (1)	0.2392 (4)
6	0.3232 (5)	0.2764 (2)	0.2772 (3)	0.2683 (1)	0.3013 (4)
7	0.3460 (5)	0.3047 (2)	0.3065 (3)	0.3002 (1)	0.3352 (4)
8	0.2816 (5)	0.2453 (3)	0.2443 (2)	0.2375 (1)	0.2673 (4)
9	0.3492 (5)	0.3100 (3)	0.3066 (2)	0.3021 (1)	0.3300 (4)
10	0.3124 (5)	0.2680 (2)	0.2706 (3)	0.2645 (1)	0.2941 (4)
11	0.2932 (5)	0.2433 (2)	0.2434 (3)	0.2367 (1)	0.2693 (4)
12	0.2947 (5)	0.2502 (3)	0.2478 (2)	0.2426 (1)	0.2736 (4)
13	0.2818 (5)	0.2368 (2)	0.2377 (3)	0.2303 (1)	0.2657 (4)
Avg. rank pos.	*5.00*	*2.38*	*2.62*	*1.00*	*4.00*

some features (otherwise, we have had to sample the set of items to randomly chose the sensors, and we could not have used the $\boldsymbol{u}^{(\mathrm{w}\odot\mathrm{uti})}$ encoding).

6.2.2 Results and discussion

In Table 6, we report the results achieved for five out of the seven user encodings presented in Section 4. We cannot use neither $\boldsymbol{u}^{(\mathrm{rate})}$ nor $\boldsymbol{u}^{(\mathrm{int})}$ mainly because of the extremely high sparsity of the user-item matrix, with almost no items in common between users. Moreover, the number of different items in the datasets will pose a serious issue regarding the size of the vectors to represent users.

If we rank the encodings with respect to the scores obtained in each dataset (each day represents a dataset), we observe that the $\boldsymbol{u}^{(\mathrm{w}\odot\mathrm{uti})}$ encoding is always in the first position of the ranking, yielding the lowest values of Q_{loss}, while the $\boldsymbol{u}^{(\mathrm{raw})}$ description of users is always in the 5^{th} position, with the worst results. The differences in ranking scores of the $\boldsymbol{u}^{(\mathrm{w}\odot\mathrm{uti})}$ representation and the other methods are significant with p-value=0.05 using the Bergmann's test (as proposed by [10]).

The average ranking of $\Omega(\mathcal{U}^{(\mathrm{uti})})$ is 2.62, while the hybrid $\Omega(\mathcal{U}^{(\mathrm{raw}\oplus\mathrm{uti})})$ reaches slightly better ranking, 2.38, although the difference is not statistically significant. However the quality of both clusterings is significantly better (with p-value=0.05) than clustering $\Omega(\mathcal{U}^{(\mathrm{emb})})$.

As previously mentioned, the $\boldsymbol{u}^{(\mathrm{w}\odot\mathrm{uti})}$ representation can only be calculated when the sensors (representative items) are obtained through a cluster of items, as has been done in this experiment. If the sensors were taken randomly from the set of items, then the best representation we could have would be $\boldsymbol{u}^{(\mathrm{uti})}$, because the quality of $\Omega(\mathcal{U}^{(\mathrm{uti})})$ is similar to that of $\Omega(\mathcal{U}^{(\mathrm{raw}\oplus\mathrm{uti})})$ and it has proven to be better in the benchmark results of Section 6.1.3.

The $\boldsymbol{u}^{(\mathrm{raw})}$ encoding is always the worst, surely motivated by the scarce information of the users: only the platform (type of device used for browsing and reading) and its GPS coordinates. If there were more and better user information, maybe the quality of the $\Omega(\mathcal{U}^{(\mathrm{raw})})$ clustering could have been improved, although we do not expect it to outperform the rest of the encodings because the users' characteristics are not normally related to their individual preferences.

7 Conclusions

We have presented a method to assess the quality of clusters of users in the context of recommender systems. This method can be applied in any recommendation dataset, but it is especially relevant for those problems in which the user-item matrix has extremely few data. The method relies on an entropy-based measure which is computed using a probabilistic estimation of the preferences of the users in the cluster with respect to a reduced group of representative items (sensors).

Inspired on the quality measure, we have proposed some alternatives to encode users employing also the probabilistic estimation of their tastes. The estimation is obtained with a utility function, learned using a recommender system.

We tested our proposal using a typical matrix factorization based recommender system to learn the utility function on six public domain datasets, as well as a preference learning based RS on an additional problem which comprises 13 very large datasets with an extremely sparse user-item matrix.

The experiments let us conclude that it is feasible to apply our methodology in very large datasets. Moreover, the results lead us to advocate for a utility-based encoding of users in the context of recommender systems, when our aim is to obtain homogeneous clusters with respect to the preferences of users. More specifically, we advocate for the use of the $\boldsymbol{u}^{(\mathrm{w}\odot\mathrm{uti})}$ encoding as long as the characteristics of the items are available, and for the $\boldsymbol{u}^{(\mathrm{uti})}$ encoding otherwise.

Finally, we would like to highlight that it is possible to take advantage of the duality between users and items. Therefore, the methodology presented in this paper to cluster users can be easily adapted to cluster items based on the preferences that users have over them (in this case, it would be necessary to select a set of representative users who would act as sensors). This can be interesting to find groups of items (products, services, etc.) very heterogeneous with respect to their characteristics, but very homogeneous regarding the kind of user that could be eventually interested in them.

Acknowledgements We are grateful to Outbrain for providing the dataset used in the paper, and to NVIDIA Corporation for the donation of the Titan Xp GPU used in this research. This work was funded under grant PID2019-109238GB-C21 from the Spanish Ministry of the Economy and Competitiveness, and IDI-2018-000176 from the Principado de Asturias Regional Government, partially supported with ERDF funds. Pablo Pérez-Núñez acknowledges the support of the Principado de Asturias Regional Government under Severo Ochoa Program (ref. BP19-012).

Funding Open Access funding provided thanks to the CRUE-CSIC agreement with Springer Nature.

References

1. Abadi M, Barham P, Chen J, Chen Z, Davis A, Dean J, Devin M, Ghemawat S, Irving G, Isard M, Kudlur M, Levenberg J, Monga R, Moore S, Murray DG, Steiner B, Tucker P, Vasudevan V, Warden P, Wicke M, Yu Y, Zheng X (2016) Tensorflow: A system for large-scale machine learning. In Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2016
2. Ali Z, Kefalas P, Muhammad K, Ali B, Imran M (2020) Deep learning in citation recommendation models survey. Exp Syst Appl, 113790
3. Bahamonde A, Bayón GF, Díez J, Quevedo JR, Luaces O, del Coz JJ, Alonso J, Goyache F (2004) Feature subset selection for learning preferences: A case study. In Proceedings of the International Conference on Machine Learning (ICML '04) (Banff, Alberta (Canada)), R. Greiner and D. Schuurmans, Eds., pp. 49–56
4. Boratto L, Carta S (2011) State-of-the-Art in Group Recommendation and New Approaches for Automatic Identification of Groups. In: Soro A, Vargiu E, Armano G, Paddeu G (eds) Information Retrieval and Mining in Distributed Environments. Springer, Berlin Heidelberg, pp 1–20
5. Bu J, Shen X, Xu B, Chen C, He X, Cai D (2016) Improving Collaborative Recommendation via User-Item Subgroups. IEEE Trans Knowl Data Eng 28(9):2363–2375
6. Dara S, Chowdary CR, Kumar C (2020) A survey on group recommender systems. J Int Info Syst 54(2):271–295
7. Dau A, Salim N (2020) Recommendation system based on deep learning methods: a systematic review and new directions. Artif Intell Rev 53(4):2709–2748
8. Fan B, Zhang P (2009) Spatially enabled customer segmentation using a data classification method with uncertain predicates. Decis Support Syst 47(4):343–353

9. Fang H, Zhang D, Shu Y, Guo G (2020) Deep learning for sequential recommendation: Algorithms, influential factors, and evaluations. ACM Transactions on Information Systems (TOIS) 39(1):1–42
10. Garcia S, Herrera F (2008) An extension on statistical comparisons of classifiers over multiple data sets for all pairwise comparisons. J Mach Learn Res 9:2677–2694
11. Goldberg K, Roeder T, Gupta D, Perkins C (2001) Eigentaste: A constant time collaborative filtering algorithm. Inf Retr 4(2):133–151
12. Gong S (2010) A collaborative filtering recommendation algorithm based on user clustering and item clustering. J Soft 5(7):745–752
13. Guo G, Zhang J, Thalmann D, Yorke-Smith N (2014) Etaf: An extended trust antecedents framework for trust prediction. In Proceedings of the 2014 International Conference on Advances in Social Networks Analysis and Mining (ASONAM)
14. Guo G, Zhang J, Yorke-Smith N (2013) A novel bayesian similarity measure for recommender systems. In IJCAI, pp. 2619–2625
15. Gupta U, Patil N (2015) Recommender system based on Hierarchical Clustering algorithm Chameleon. In 2015 IEEE International Advance Computing Conference (IACC), IEEE, pp. 1006–1010
16. Harper FM, Konstan JA (2015) The movielens datasets: History and context. ACM Trans. Interact. Intell. Syst. 5:4
17. Herbrich R, Graepel T, Obermayer K (1999) Support vector learning for ordinal regression. In Proceedings of the Ninth International Conference on Artificial Neural Networks (Edinburgh, UK), pp. 97–102
18. Herlocker JL, Konstan JA, Borchers A, Riedl J (1999) An algorithmic framework for performing collaborative filtering. In Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (New York, NY, USA), SIGIR '99, Association for Computing Machinery, p. 230-237
19. Hiziroglu A (2013) Soft computing applications in customer segmentation: State-of-art review and critique. Expert Syst Appl 40(16):6491–6507
20. Jiang S, Ding Z, Fu Y (2020) Heterogeneous recommendation via deep low-rank sparse collective factorization. IEEE Trans Pattern Anal Mach Intell 42(5):1097–1111
21. Joachims T (2002) Optimizing search engines using clickthrough data. In Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD), pp. 133 – 142
22. Katarya R, Verma OP (2017) An effective web page recommender system with fuzzy c-mean clustering. Multi Tools Appl 76(20):21481–21496
23. Kingma D, Ba J (2014) Adam: A method for stochastic optimization. arXiv preprint https://arxiv.org/abs/1412.6980
24. Koren Y, Bell R, Volinsky C (2009) Matrix factorization techniques for recommender systems. Computer 42(8):30–37
25. Liao CL, Lee SJ (2016) A clustering based approach to improving the efficiency of collaborative filtering recommendation. Electron Commer Res Appl 18:1–9
26. Liu J, Wu C (2017) Deep learning based recommendation: A survey. In International Conference on Information Science and Applications, Springer, pp. 451–458
27. Luaces O, Díez J, Joachims T, Bahamonde A (2015) Mapping preferences into euclidean space. Expert Syst Appl 42(22):8588–8596
28. Lucas JP, Laurent A, Moreno MN, Teisseire M (2012) A fuzzy associative classification approach for recommender systems. Internat J Uncertain Fuzziness Knowledge-Based Systems 20(04):579–617
29. O'Connor M, Herlocker J (1999) Clustering items for collaborative filtering. Proceedings of the ACM SIGIR Workshop on Recommender Systems
30. Park YJ (2013) The adaptive clustering method for the long tail problem of recommender systems. IEEE Trans Knowl Data Eng 25(8):1904–1915
31. Park YJ, Tuzhilin A (2008) The long tail of recommender systems and how to leverage it. Proceedings of the 2008 ACM conference on Recommender Systems RecSys 08, 11–18
32. Reddy S, Labutov I, Joachims T (2015) Learning representations of student knowledge and educational content. In Proceedings of the 31st ICML (Workshop: Machine Learning for Education)
33. Reddy S, Labutov I, Joachims T (2016) Learning student and content embeddings for personalized lesson sequence recommendation. In Proceedings of the Third (2016) ACM Conference on Learning@ Scale, ACM, pp. 93–96
34. Roy S, Biswas M, De D (2020) imusic: a session-sensitive clustered classical music recommender system using contextual representation learning. Multi Tools Appl 79(33):24119–24155
35. Sculley D (2010) Web-scale k-means clustering. In Proceedings of the 19th international conference on World wide web, ACM, pp. 1177–1178

36. Selvi C, Sivasankar E (2019) A novel adaptive genetic neural network (agnn) model for recommender systems using modified k-means clustering approach. Multimedia Tools and Applications 78(11):17763–17798
37. Shannon CE (1948) A mathematical theory of communication. Bell Syst Tech J 27(3):379–423
38. Wu RS, Chou PH (2011) Customer segmentation of multiple category data in e-commerce using a soft-clustering approach. Electron Commer Res Appl 10(3):331–341
39. Wu Y, Liu X, Xie M, Ester M, Yang Q (2016) CCCF: Improving Collaborative Filtering via Scalable User-Item Co-Clustering. Proceedings of the Ninth ACM International Conference on Web Search and Data Mining - WSDM ’16, 73–82
40. Xu B, Bu J, Chen C, Cai D (2012) An exploration of improving collaborative recommender systems via user-item subgroups. Proceedings of the 21st international conference on World Wide Web - WWW ’12, 21
41. Zhang S, Yao L, Sun A, Tay Y (2019) Deep learning based recommender system: A survey and new perspectives. ACM Comput Surv 52:1
42. Ziegler CN, McNee SM, Konstan JA, Lausen G (2005) Improving recommendation lists through topic diversification. In Proceedings of the 14th International Conference on World Wide Web (New York, NY, USA), WWW ’05, Association for Computing Machinery, p. 22-32

Publisher’s note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.