Information Sciences 536 (2020) 156–170

Contents lists available at ScienceDirect

Information Sciences

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

New technique to alleviate the cold start problem in recommender systems using information from social media and random decision forests

J. Herce-Zelaya [a], C. Porcel [b,*], J. Bernabé-Moreno [a], A. Tejeda-Lorente [a], E. Herrera-Viedma [a,*]

[a] Department of Computer Science and Artificial Intelligence, University of Granada, Granada, Spain
[b] Department of Computer Science, University of Jaén, Jaén, Spain

ARTICLE INFO

Article history:
Received 7 November 2019
Received in revised form 12 May 2020
Accepted 18 May 2020
Available online 28 May 2020

Keywords:
Recommender systems
Cold-start problem
Social media
Decision tree classifier
Random forest

ABSTRACT

The aim of recommender systems is to provide users with items that could be of their interest. However one of the biggest drawbacks from recommender systems is the so called cold start problem, which occurs when new users or products are added to the system and therefore there is no previous information about them. There are many proposals in the literature that aim to deal with this issue. In some cases the user is required to provide some explicit information about them, which demands some effort on their part. Because of that and due to the great boom of social networks, we will focus on extracting implicit information from user's social stream. In this paper we will present an approach on which social media data will be used to create a behavioural profile to classify the users and based on this classification will create predictions making use of machine learning techniques such as classification trees and random forests. Thus the user will not have to provide actively any kind of data explicitly but their social media source, alleviating in this way the cold start problem since the system would use this data in order to create user profiles, which will be the input for the engine of the recommender systems. We have carried out numerous experiments, as well as a comparison with some other state-of-the-art new user cold-start algorithms, obtaining very satisfactory results.

1. Introduction

As years go by the amount of content that we can find in the Internet is exponentially growing. This enables users to find any type of content. Nevertheless it also adds another implicit problem: the difficulty to find relevant items for a determined user [14]. The more content we have, the more arduous is to spot the relevant items for the users out of the total amount of content. Recommender systems palliate this problem by helping us making decisions or finding what we are looking for [5]. Recommender systems automate the process of recommendation that we follow in our daily life, by asking for opinion to other users [10,26].

In order to accomplish that, these systems utilize previous behaviours and analogies between users to predict new demands or preferences. Recommender systems have been implemented in numerous fields, mainly e-commerce

* Corresponding author.
E-mail addresses: julioherce@gmail.com (J. Herce-Zelaya), cporcel@ujaen.es (C. Porcel), juan.bernabe@webentity.de (J. Bernabé-Moreno), atejeda@decsai.ugr.es (A. Tejeda-Lorente), viedma@decsai.ugr.es (E. Herrera-Viedma).

https://doi.org/10.1016/j.ins.2020.05.071
0020-0255/

[9,12,36], but also in many others, such as university digital libraries [37,40,41], or in educational field [17]. In fact, these systems have a crucial role in highly rated Web sites, such as Amazon,[1] YouTube,[2] Netflix,[3] Tripadvisor,[4] Last.fm[5] or IMDb[6] [12].

Many recommendation proposals utilize data provided by the user in order to be able to recommend items or products to them. This data could be acquired either in a explicit (e.g. by directly rating a product) or implicit way (e.g. by establishing a connection with another user). However, when a user first joins a site, they have not yet expressed their interest about any product: they do not have any ratings history. Due to these circumstances it is hard to infer what the users are going to like when they start on a site. The problem is referred to as the cold-start problem [4,18,48]. Another case of cold-start problem would be when a new product is released and therefore there are no data for this product since no one could yet rate it. These problems are known as new user cold-start problem and new item cold-start problem. As an example of the new item cold-start problem, we could imagine a web site which provides streaming media and video-on-demand. This site lacks of any data from the user that just joined and they do not have any hint on what such users may like. Therefore it can not be established whether certain movie is suited to be recommended for the user or not. These kind of sites are aware of these issues and in order to address them, they often offer to the users the opportunity to make a voting tour in where they are asked to rate a couple of items before the site can recommend others to them. Other sites tend to ask users to give some info about them, such as interests or hobbies [33].

One drawback from these methods is that they require some explicit action from the user and they tend to be reluctant to make these movie rate tours or give more info about them. Nowadays there are a vast amount of different movie streaming services such as Netflix, Movistar, HBO, Filmin, Sky, or Watchever, and it would be a very laborious task for the users to make these rate tours or filling up the extra info about them every time they are trying a new service. These services often offer a free trial first month of service and therefore the time the user arrived is the most crucial time since user will not stay if the do not like the service offering. Thus it is even more decisive for them to show to the user relevant content that the user would like to see as soon as possible convincing them to stay.

Despite of the reticence from users in general to fill up data forms and to make these rating tours in that kind of situation, there are other contexts where these users can actively provide more information about their likes, tastes and behaviours. We refer here indeed to the social media systems [2,46]. Nowadays users leave in microblogs and social media systems a huge amount of information about their interest expressing their tastes and opinions [3,42,46]. Social media systems are in general an environment where users tend to express themselves in a broad way leaving an enormous digital footprint that could be converted in valuable information [1]. Due to the ever-growing usage of social media and microblogs, used for keeping in touch with people as well as for expressing their opinion about very different topics, a vast amount of information could be extracted and converted into useful knowledge in order to integrate them into a recommender system and generate better estimates [18,26,42,49]. Therefore, this user's social content could be used and processed with the purpose of elaborating a user's profile that could help the decision making process [1,3]. This procedure reduces significantly the users' interaction since we do not require much of actively action from them which eases the required flow previous to the recommendation process.

In this paper, our main target is to create a behavioural profile leveraging these implicit data extracted from the user's social stream [18,27,28] which would be determined by a collection of features depicting their preferences, tastes and character. We have corroborated that in previous proposals those approaches have worked satisfactorily [35,39]. However our proposal provides the novelty of being able to bring those topics to the practical area since we provide a data set merging two data sources and matching them together in order to be able to bind the social stream data with the rating data. After having a behavioural mapping for the users, the different attributes from their profiles will be used to establish a classification matching the behavioural profiles with the target attribute, which is the user's rating for a determined item. A prediction model will be created for every item from the catalogue in order to predict whether such item should be recommended to a determined user or not. Concretely, the concept is to extract information provided on the most well-known and used microblog system, i.e., Twitter[7] and use this knowledge in the recommendation scheme, an integration that is giving increasingly better results [11,24,42]. Then this knowledge is linked with some rating data in order to create the models.

However, the amount of information we have to deal with in this type of processes is so massive that it is necessary to resort to more advanced additional techniques that could help us to obtain useful knowledge. Therein we propose to integrate in this process some machine learning techniques, since they are the techniques that recently have been utilized with most success in recommendation systems [28,30,31]. To assist the processing of massive data, we will adopt Big Data tools in order to give better recommendations [27]. Specifically we propose to use Apache Spark[8] which is a fast and general engine for large-scale data processing that offers a broad diversity of machine learning algorithms. Our approach will be based on the

[1] http://www.amazon.es/.
[2] www.youtube.com/.
[3] www.netflix.com/.
[4] www.tripadvisor.es/.
[5] www.lastfm.es/.
[6] www.imdb.com/.
[7] https://twitter.com/.
[8] https://spark.apache.org/.

use of decision trees and random decision forests that will assist us for the classification of the users according to their profiles, allowing us to obtain significantly better recommendations [22].

Then, the starting point of our proposal is a model to process the social stream for every single user [15,16,28]. From this stream we will extract some features creating the so called *Twitter Profile* which defines the behaviour of the users of the social media site. Then this Twitter profile is utilized to classify the users in relation with the actual rating of every item (which would be the target), with the assist of decision trees (either single decision tree or random forest). After this classification process we will select the items which are predicted to be recommended for the user with the highest probability. We have developed the proposed model and, to check the functionality of the system, we have applied it in the movies recommendation field [11]. We have selected the field of movies because the data is more accessible and it is a topic in which an extensive amount of people is interested and thus will be easier to find people for whom the movies are among their interests. Furthermore, movies are a very common and active topic of discussion in the social networks. The results achieved after the validation and experimentation phase indicate a favourable behaviour of the proposed model, being therefore very suitable for its application in a real environment.

The rest of the paper is structured as follows. In Section 2, the necessary preliminaries are exposed. Next in Section 3, we describe our proposal. Thereafter, in Section 4 we describe the experiments and evaluation of the system. Finally, some closing remarks and future works are pointed out in Section 5.

2. Preliminaries

2.1. Basis of recommender systems

Recommender systems are information filter tools that aids users in their information access processes, through prediction and recommendations of information items that could be from the user's concern [5,10]. This process is offered as an alternative to ordinary social recommendation process. Namely, the traditional process in which we all follow by requesting opinions from experts or acquaintances when we have to make a choice for acquiring a new product without possessing any information about the product itself.

Recommender systems are not just a search method, they go beyond the search. They do not respond to punctual information needs, on the contrary they bring us deep into the discovery web. The users do not search for something specific anymore, but they expect to discover things that they did not even know that exist, or things they did not know how to execute the search to find them. The problem that these systems intent to solve is to expose items which are unknown by the user. In order to achieve that, the system will have to execute unknown ratings estimation methods, using known ratings that are persisted in a rating matrix. We could classify this estimation methods in two types [26]:

- Model based: Initially they develop a model through machine learning techniques and afterwards the model is queried in order to provide some useful recommendations about items. For instance, some techniques that could be used are clustering, neural networks or decision trees.
- Heuristic or memory based: Systems that provide recommendations making use of heuristic formulas of similarity and correlation between items and user. Example of these similarity measures are: Pearson or cosine similarity.

In this proposal we focus on the first type, namely we propose a model-based approach.

2.2. Cold start problem

As previously mentioned, ratings are estimated making use of a rating matrix. The complication here is that users tend to be reluctant to provide personal information. Thus in real applications these rating matrices tend to be disperse, what complicates the process of estimating recommendations since we can not access the past historical from user nor from items. This situation is what is defined as a cold start problem [18,23].

This problem appears inevitable in two situations:

- New users cold start problem [4]: It appears when a new users is joined to the system, because they have not supplied any information yet and therefore, it can not be recommended with any item nor be compared with any similar user to them.
- New items cold start problem [35,45]: It appears when a new item is added to the system. Since there are no ratings about this item, it will not be chosen in any matching process and thus it will not be recommended to any user.

2.3. Decision tree classifier

Decision tree classifier learning is a machine learning algorithm that is often utilized in decision making by generating predictions for labelled data using as input a series of observations. The goal is to create a model that predicts the value of a target variable, i.e., label, based on several input variables. They are able to discover complex interactions between variables and create accurate predictions on new data.

There are two main kind of decision trees according to the nature of their outcome, which are the following:

- Classification tree analysis is when the predicted outcome can take a discrete set of values. Commonly the response variable has two categories, e.g. yes or No. If there are more than two categories then the algorithm C4.5 will be used.
- Regression tree analysis is when the predicted outcome is continuous or numeric, e.g. the price of a car, the amount of rain to fall.

Since we want to figure out whether a movie is suitable to be recommended to a user or not, our outcome will be categorical with only two categories, recommended and not recommended. Therefore in our proposal we use a classification tree.

The term Classification And Regression Tree (CART) analysis is an term which alludes to both of the previously described techniques. First introduced by Breiman et al. [8], trees used for regression and trees used for classification have some analogies but also some differences, such as the procedure used to determine where to split.

Decision tree builds classification or regression models in the form of a tree structure. They break down a dataset into smaller and smaller subsets and simultaneously a decision tree is being developed. At the end we have as result a tree with decision nodes and leaf nodes. A decision node (e.g. outlook) has two or more branches (e.g. sunny and rainy). Leaf node (e.g., Play tennis) represents a classification or decision. The best predictor will be positioned at the top node and it is called root node. Decision trees can deal with both categorical and numerical data.

In Table 1 is displayed an example of observations for building a classifier tree. The observations' target is *play tennis* and the features are outlook whose categories are rainy or sunny; humidity with categories of high and low; and windy with categories of false or true.

If we calculate the decision tree for the previous values we will obtain a classified tree like the one in Fig. 1. As described before every node represents a decision. The algorithm utilized to create the tree is known as ID3 and was proposed by J.R. Quinlan in [32].

2.4. Random forest

Single decision trees tend to suffer from high variance or high bias. Random forests come as an attempt to mitigate the problems that these high variance or high bias might cause. Random Forests are an ensemble learning method for classification and regression whose mechanism is based on building multiple decision trees at training time and processing their results in order to get a more stable prediction out of them. For classification, it selects the class that is the mode of the classes output by the individual trees, while for regression, the mean of different regression trees will be calculated [7].

The previously described process defines the original bagging algorithm for trees [6]. Random forests have one difference with this general scheme: they make use of a slightly modified tree learning algorithm that selects, for every decision tree in the learning process, a random subset of the features. This process is called *feature bagging*. This is done because of the correlation of trees in an ordinary bootstrap sample: if one or a few features are very strong predictors for the response variable (target output), these features will be selected in many of the other trees, causing them to become correlated. With that it is intend to add a random aspect to the algorithm in order to prevent correlation between the different classification trees [21].

Then, random decision forests are a combination of tree predictors: the basic principle from random forest is to combine some *weak learners* in order to obtain a *strong learners*. For this reason, they are a useful tool for making predictions taking into account that they do not overfit because of the law of large numbers. Embracing the right amount and kind of randomness to the algorithm makes them accurate classifiers and regressors.

2.5. Related works

We can find in the literature diverse proposals that aim to solve the cold start problem [18,28]. A possible workaround for solving that problem is to demand users to provide some personal info and to rate a determined number of items in order to be able to establish a profile in the system [19]. If we focus on the new users cold start problem, in [23] a comparative study from different proposals is proposed, some of them will be cited here. In particular, in [4] the authors introduce a new optimized similarity measure through machine learning techniques based neural networks. In [44] is proposed an approach that incorporates association rules, probability based metrics and own users' context in order to alleviate the cold start problem. In [20] a rules based system is also utilized, in this case a probabilistic model. Nevertheless, in [25] a proposal is presented in which classification algorithm C.4.5 and Naive Bayes are combined with diverse similarities and prediction techniques in order to solve the problem. Another interesting proposal is introduced in [13], where they present an approach that utilizes trust and distrust network to find trustworthy users and utilize the suggestion of these users to generate the recommendations. In this sense of fetching social network data to palliate this problem, more recently, in [18] is presented a revision about how it is been working precisely with information extracted from social network, studying some published articles between 2011 and 2017.

On the other hand, in order to deal with the new items cold start problem there are not so many articles to be found, although we could mention two interesting ones. In [45] the authors present a system in which they obtain items features using deep learning architecture SDAE and those features are exploited by integrating them in collaborative schema timeSVD

Table 1
Observations.

Outlook	Humidity	Windy	Play tennis
Rainy	High	True	No
Sunny	Low	True	Yes
Sunny	High	False	No
Rainy	Low	False	No
Sunny	High	False	Yes
Rainy	High	False	No
Sunny	Low	True	No

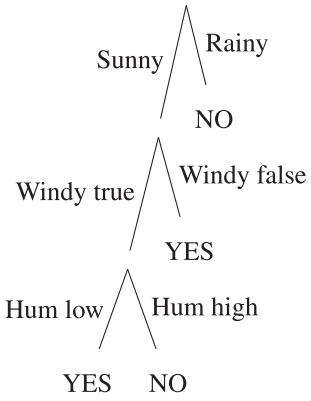

Fig. 1. Classifier tree.

++. Furthermore, in [35] items features and user's generated tags are used applying a further matrix factorization steps is used in order to generate the knowledge.

There are many other proposals that allow to alleviate the problem with their approaches, although they are not specific for that. In general, we can see that one of the most used and effective techniques is to utilize additional info about users, such as age, gender, education, zip code or any other information that could help to classify the users, which is what is known as demographic information [29].

Other works focus their approach on how the recommender systems can be empowered and developed through machine learning techniques and how challenging is to find the most suitable technique for every use case, reviewing the trends of machine learning and artificial intelligence techniques and identify open questions in the use or research of machine learning algorithms [31]. Other machine learning techniques like deep learning are presented in a broad study [43]. There is also in the literature some works that study the use of based social recommender systems making use of the data from social networks to improve the accuracy of the algorithms [47]. In other works like [34] they also use external behavioural data to fill the gaps between the human and software decision making process. We can find as well some works that aims to solve the cold-start problem from different approaches, as in [38] where we can find a comprehensive review of different studies and a extensive comparative between them.

If we focus on the idea of incorporating information extracted from social media and social networks into the recommendation schemes, in [48] an approach is presented where social relations and temporal informations are integrated to palliate the cold-start problem making use of Markov Chains. In [16] the authors propose an ontology-based advertisement recommendation system that leverages the data produced by users in social networking sites; a shared ontology model is used to represent both users' profiles and the content of advertisements. In [15] the social context is also taken into account to propose a recommendation approach that presents a personalized list of tourist attractions for each tourist, based on the similarity of users' desires and interests, trust, reputation, relationships, and social communities.

3. Description of the proposal

We propose a recommender system in which the solely input is the social stream from the user. In our specific case, we will use data extracted from Twitter[9] to generate the user profiles. Then we will classify these profiles and with those data we will establish a prediction model for every item from a determined catalogue in order to predict whether such item is suitable to be recommended to that user or not. In order to do this, we propose to use decision trees that will help us to classify the users

[9] https://www.twitter.com.

according to their profiles, allowing us to obtain far better recommendations [22]. We propose to use single decision trees as also random forest, where several decision trees are ensemble in order to provide a better result [7]. In this section we explain the proposed model that we have applied to a specific environment, which is movie recommendation, and therefore we will explain it considering that specific area.

The input for our system will be the social stream from the users. To be more specific, we will use data from the user's Twitter account in order to create a Twitter profile that represents somehow the personality of the users. Afterwards we will create the prediction models with which we want to classify users according to their Twitter profile and whose target will be whether or not a particular movie is suitable or not to be recommended to certain user. In Fig. 2 we can see a diagram of the system developed and implemented, consisting of the following phases:

- Data gathering: we will first have to collect the data with which we will generate our recommendations.
- Classification: we will classify the users according to their tastes, character and behaviour (Twitter profile) making use of classification trees and random forest, whose target will be a flag indicating whether a movie is recommended or not. That will be the training process, in which we will use part of the data set to train our model. In this process the connections between the different features from Twitter profile and the target label will be created.
- Prediction: Once the models are trained, we will use the test data in order to create the predictions. Since we have also the real data from the users (movie ratings) we will be able to compare the predictions and real data evaluating so the accuracy from our predictions. At the end from this stage we will have a list of movies (the ones that we are used in our movie catalogue) and a corresponding flag for every movie indicating whether our system marked the movie as recommendable for the user or not.
- Selection: We will select all the movies that are marked as recommendable for the user and then select the ones whose probability higher is. This means, we select the movies that are marked as recommended with more certainty from our prediction model.

Then, the first challenge is to find suitable data that allows us to develop and implement our proposal. Since we need to find users who have movie rating data as well as social stream data, we decided to use Filmaffinity[10] in order to have relevant data. We will keep the focus in our proposal on movie recommendations since we find a very popular topic in social networks and also because we found a very attractive and useful tuple of source of data (Twitter for defining the users and Filmaffinity for getting the ratings that the users are giving to the movies).

Filmaffinity is an online web site that serves as movie database and where users can rate movies. The users have the possibility of provide in their Twitter url on their profile, which enables a link between both data sources. That will however diminish considerably the amount of users that we can utilize for this experiments since only roughly 10% of the users provide on their profiles their Twitter url.

We will try to predict which products are more suitable to be recommended to a user. In order to accomplish that goal, we will have a product catalogue (movie catalogue in our case), a list of users for which we have both Twitter and Filmaffinity profile. From these users we have a list of all tweets available in Twitter and a list of movie rating.

In order to classify the users depending on their Twitter profile and in relation with every movie from our movie catalogue, we will use Apache Spark. The decision of choosing is based in its ability to process large amounts of data (as is the case that concerns us), its execution speed and its machine learning library which offers a wide variety of algorithms and utilities.

To be more specific we will use the Decision Tree Classifier[11] algorithm from Spark to classify the users according to their Twitter stream and we will create prediction models for every movie in our movie catalogue. Moreover, we will also use the Random Forest Classifier[12] in order to implement the classifier with random forest. Once we have collected all the predictions, we will choose for every user those products which are predicted to be recommended and from this set we will choose those for which their probability is higher.

3.1. User data gathering

We have used an ethical web scraping in order to get the data of the users (ratings and Twitter url) from Filmaffinity. We have implemented our web scrap utilizing python as programming language. To be precise, we have used the libraries requests[13] in order to fetch the content of every page and lxml[14] in order to parse the data from every page. For persisting our data we have used a MongoDB[15] database. The reasons for choosing it are the ease to use, the fact that it has no schema and its very powerful aggregate pipeline which can effortlessly process our data records and return computed results.

[10] https://www.filmaffinity.com/es/main.html.
[11] https://spark.apache.org/docs/2.0.0/api/java/org/apache/spark/ml/classification/DecisionTreeClassifier.html.
[12] https://spark.apache.org/docs/latest/api/java/index.html?org/apache/spark/ml/classification/RandomForestClassifier.html.
[13] http://docs.python-requests.org/en/master/.
[14] http://docs.python-guide.org/en/latest/scenarios/scrape/.
[15] https://www.mongodb.com/what-is-mongodb.

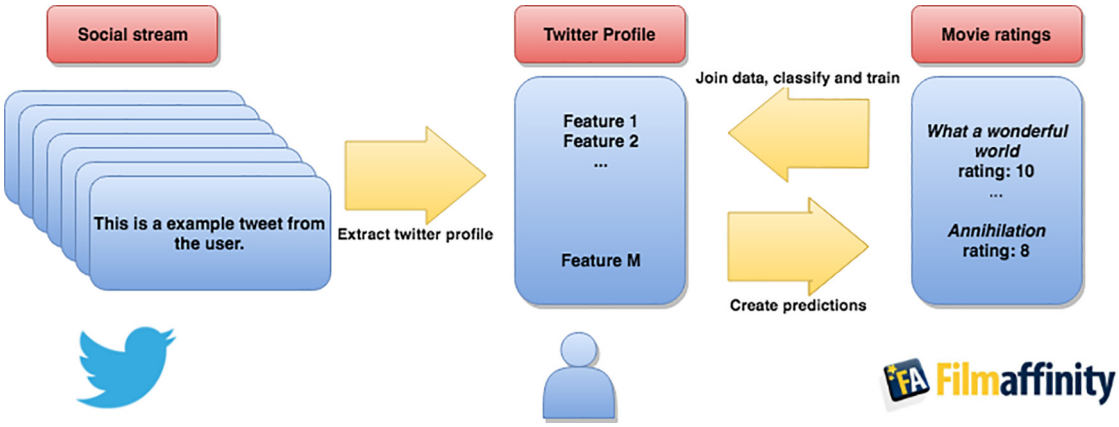

Fig. 2. Data source diagram.

We are solely interested in the subset of users that have given a Twitter url in their Filmaffinity profiles, we will discard the rest of them (since we can not match the two sources of data for them). Once we have this subset of users, we will use another ethical web scraping for getting the ratings from every of these users.

3.2. Social media data gathering

Making use of the Twitter API[16] we have collected all the tweets (actually only 3200 last tweets from every user are possible to fetch because of Twitter API's restrictions) from the users for which we have data in Filmaffinity. This data are persisted in our MongoDB database as well so that is more convenient for us to aggregate it afterwards.

3.3. Elaborating a Twitter profile

We have processed the tweets previously collected extracting features in order to build a Twitter profile for every user. This profile is defined by a series from *features* that we use later on in order to build our model.

Some of the features are directly accessible from Twitter profile, e.g. number of followers, or account creation year. However there are other features for which we need to process all the tweets in order to be able to extract them, e.g. preferred day for writing tweets, early tweeter.

The features extracted for this first iteration are the following:

- Account creation year (values: numeric)
- Early bird: Usually tweets in the morning (values: false or true)
- Night owl: Usually tweets in the night (values: false or true)
- Preferred hour: The hour on which the user more often tweets (Values from 0–23)
- Weekend tweeter: Usually tweets on weekends (values: false or true)
- Week tweeter: Usually tweets on week days (values: false or true)
- Preferred weekday: The day of the week on which the user more often tweets (Values from 0–6)
- Friends count: The number of people the user follows (values: numeric)
- Followers count: The number of followers from user (values: numeric)
- Favourites count: The number of favorited tweets by user (values: numeric)
- Geolocation enabled: If user has geolocation enabled (values: false or true)
- Number of tweets: The number of tweets from user (values: numeric)

On the other side we will have for every user a list of movie ratings. The data are those described below:

- movieId: The id of the movie in Filmaffinity.
- rating: The rating that a given user has given to the movie with id movieId.

3.4. Joining Twitter profile and movie data

As we have mentioned, we utilize the features extracted from Twitter profile in order create predictions for the rating of movies. We take the user's ratings for the movies that we have in our catalogue. The ratings are interval-based ratings drawn

[16] https://developer.twitter.com/en/docs/api-reference-index.

in a 10-point scale, that is, the ratings are integer numbers from 1 to 10. Since for us is not so important the exact rating from a user but the fact that the user liked the movie, we will convert this interval-based to a binary rating by labelling this data in two possible labels: 0 for not suitable for the user and 1 suitable for the user. We label ratings between 0 and 6 as not suitable and ratings between 7 and 10 as suitable.

For that we use a determined set of N movies, simulating a streaming service catalogue. In order to have more rating data we chose the N movies most rated from our user group (the users for which we have Twitter profile).

Then we utilize the decision tree learning algorithm[17] in order to classify the users according to their Twitter profile and as label we use a recommended flag whose values could be 0 or 1. Values with 0 means that a movie is not suitable to be recommended to a determined user and on the contrary values with 1 means that movie is suitable to be recommended to the user.

Since we want to know the predictions for several movies and not just one, we will have to use a model per movie. Thus we create N models being N the number of movies in the catalogue. The difference between those models will be only the labels (the recommended flag for a determined movie). Everything else (Twitter profile) will remain identical in every model. As described in the example from Table 4, we have a list of observations (one per user who voted the movie 1) with some features (user's features extracted from Twitter profile that were previously described) and at the end we have the label which is the value that we want to predict and have two possible values: 0 would mean that the movie A is not suitable to be recommended for corresponding user; and the value 1, in the other hand, would mean that we could recommend movie A to that user.

From the rating matrix perspective, we iterate over the items and create as many models as items in the catalogue. We simplify the interval-based ratings in 10-point scale by replacing them with binary ratings. Then we incorporate the user profile for every user. After that, we train the models with the user profiles and the ratings being the target. And lastly, we test our system with the respectively trained models and evaluate the expected values with the actual ones.

We have the rating matrix:

$$R_{mn} \tag{1}$$

where:

m = number of users.
n = number of items.

And on the other side we have the user profile:

$$P_{mt} \tag{2}$$

where:

m = number of users.
t = number of features of user's profile.

We combine the ratings matrix R_{mn} with the user profile P_{mt} where m is the number of users, n the number of items and t the number of features of user's profile.

We iterate over the different items (movies) creating a different model for each starting from the rating matrix as indicated in Table 2. Afterwards we combine the ratings matrix R_{mn} with the user profile P_{mt} like indicated in Table 3.

$$R_{ij} \tag{3}$$

Our proposed model takes our training data and then creates a decision tree and a random forest per model (one model per movie as previously indicated) and from these models, it will create the corresponding predictions for the values for the test data. This data pipeline is displayed in Fig. 3.

4. Evaluation and experiments

4.1. Predictions evaluation

Once that our classification trees and random forests have created the predictions, we will continue by evaluating the quality of these predictions, exposing the results from the two different variations (single classification tree and random forest) in order to compare them. The target from our prediction model is a flag that indicates whether a movie is recommended or not. We will then select the movies that are marked as recommended and from them we select those which have higher probability to be recommended.

We will quantify the accuracy from the predictions with the following indicators. We have chosen these indicators since they are the most common metrics for evaluation Classification Trees or Random Forest, especially the precision, RMSE and

[17] http://en.wikipedia.org/wiki/Decision_tree_learning.

Table 2
For every item = j.

r_{0j}
r_{1j}
...
r_{mj}

Table 3
For every item = j.

p_{00}	p_{01}	...	p_{0t}	r_{0j}
p_{10}	p_{11}	...	p_{1t}	r_{1j}
...	...	...	...	...
p_{m0}	p_{m1}	...	p_{mt}	r_{mj}

Table 4
Model for movie A.

UserId	Feature-1	Feature-2	...	Feature-M	Label
1	6	0	–	22	0
2	7	1	–	20	1
3	1	1	–	18	0
4	2	0	–	18	0
5	0	1	–	17	1

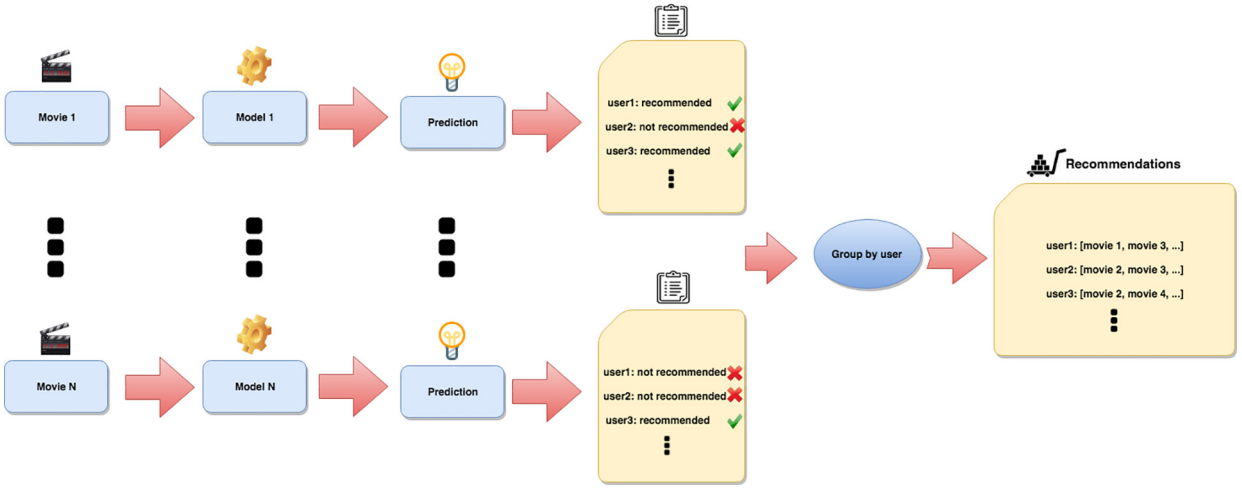

Fig. 3. Recommendation prediction diagram.

F-measure. The average rating from recommended items gives a quick and clear vision of how good the system was with their predictions.

- Average rating from recommended movies. From all the movies that are predicted to be recommended for the user we calculate the average of ratings provided about each movie. It is an indicator of, how good are in general the movies that are recommended to the user.
- Precision (positive predictive value). Resembles the percentage of success from our recommender over the failures. In the table is recorded the opposite value, i.e. accuracy error (ACC):

$$\mathrm{ACC} = \frac{TP}{TP + FP}.$$

- Root Mean Squared Error (RMSE). Represents the sample standard deviation of the differences between predicted values and observed values:

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{t=1}^{n} e_t^2}.$$

- F-measure. The F measure is a measure of the accuracy of a test. It is defined as the weighted harmonic mean of the precision and recall of the test (F1):

$$\mathrm{F1} = \frac{2 \cdot precision \cdot recall}{precision + recall}.$$

4.2. Developed experiments

With the use of web scraping techniques, we extracted from Filmaffinity a total of 8503 users from Filmaffinity web site. From all these users, we filter out those that do not provide their Twitter url in their Filmaffinity profile. That gives a total of 604 users which have a Twitter profile. After filtering out those with false Twitter url and those that doesn't have tweets in their profile we have 482 users left.

From these users we have a total of 781 782 ratings to make our experiments. That makes an average of roughly 1622 ratings per user. The total number of movies rated is 55769. On the other side, we have a total of 1142720 tweets to process, an average of 2370 tweets per user.

Once we process the Twitter stream from every user, we generated a Twitter profile for every user which would look like this sample for the movie with movie id 160882 in Table 5.

The columns from Table 5 are described as follow:

- id: User id
- f1: account year of creation
- f2: early bird
- f3: night owl
- f4: preferred hour
- f5: weekend tweeter
- f6: week tweeter
- f7: preferred weekday
- f8: friends count
- f9: followers count
- f10: favourites count
- f11: geo enabled
- f12: number of tweets

Then we interpolate the Twitter profile data with the movie rating data obtaining something like the sample from Table 6.

The meaning of the columns from f1 to f12 is the same as in the previous case; in this case we see a new column with the rating from the user to the movie.

After that, we group the observations by movie. The observation will consist of the features from the Twitter profile of every user and the target (or label) will be a flag indicating if the movie is recommended for this user. As previously stated, we consider that a movie is suitable to be recommended to a user if the user has rated it with a rating of 7 (out of 10) or more. And then our entries would look like following sample in Table 7. We can see that we do not have the rating column any-

Table 5
Twitter profile sample.

id	f1	f2	f3	f4	f5	f6	f7	f8	f9	f10	f11	f12
305054	2010	False	False	23	False	False	0	759	123	3	False	8462
990342	2013	True	False	9	False	False	2	1518	1169	27829	True	50207
982478	2010	False	True	23	False	False	2	481	477	829	True	8116
469948	2010	False	False	21	False	False	3	64	122	88	True	3516
832140	2013	True	False	11	False	False	4	805	531	2932	True	10216
547430	2013	True	False	9	False	False	2	1777	6195	19288	False	30364
106161	2011	False	True	21	False	False	0	37	48	107	True	1321
707180	2011	True	False	12	False	False	3	337	425	1623	True	9772
525484	2010	False	True	23	False	False	2	247	248	162	False	2883
541976	2011	False	True	22	False	False	2	654	585	2589	False	3832
715072	2011	False	True	12	False	False	1	467	581	830	False	16469

more, but we have now the recommended column that is directly calculated from rating and it would have the value 1 when the movie is suitable to be recommended to the user; and otherwise would have value of 0. The description of the columns from f1 to f12 is the same as in the previous two.

Now we will create a classifier tree and random forest per every movie. To do it we use a 70% of the users to train the different models and the 30% remaining to evaluate the predictions.

Table 6
Twitter profile sample with ratings.

id	f1	f2	f3	f4	f5	f6	f7	f8	f9	f10	f11	f12	rate
305054	2010	False	False	23	False	False	0	759	123	3	False	8462	5
990342	2013	True	False	9	False	False	2	1518	1169	27829	True	50207	4
982478	2010	False	True	23	False	False	2	481	477	829	True	8116	7
469948	2010	False	False	21	False	False	3	64	122	88	True	3516	5
832140	2013	True	False	11	False	False	4	805	531	2932	True	10216	9
547430	2013	True	False	9	False	False	2	1777	6195	19288	False	30364	6
106161	2011	False	True	21	False	False	0	37	48	107	True	1321	8
707180	2011	True	False	12	False	False	3	337	425	1623	True	9772	4
525484	2010	False	True	23	False	False	2	247	248	162	False	2883	7
541976	2011	False	True	22	False	False	2	654	585	2589	False	3832	5
715072	2011	False	True	12	False	False	1	467	581	830	False	16469	8

Table 7
Twitter profile sample with recommended label.

id	f1	f2	f3	f4	f5	f6	f7	f8	f9	f10	f11	f12	rec
305054	2010	False	False	23	False	False	0	759	123	3	False	8462	0
990342	2013	True	False	9	False	False	2	1518	1169	27829	True	50207	0
982478	2010	False	True	23	False	False	2	481	477	829	True	8116	1
469948	2010	False	False	21	False	False	3	64	122	88	True	3516	0
832140	2013	True	False	11	False	False	4	805	531	2932	True	10216	1
547430	2013	True	False	9	False	False	2	1777	6195	19288	False	30364	0
106161	2011	False	True	21	False	False	0	37	48	107	True	1321	1
707180	2011	True	False	12	False	False	3	337	425	1623	True	9772	0
525484	2010	False	True	23	False	False	2	247	248	162	False	2883	1
541976	2011	False	True	22	False	False	2	654	585	2589	False	3832	0
715072	2011	False	True	12	False	False	1	467	581	830	False	16469	1

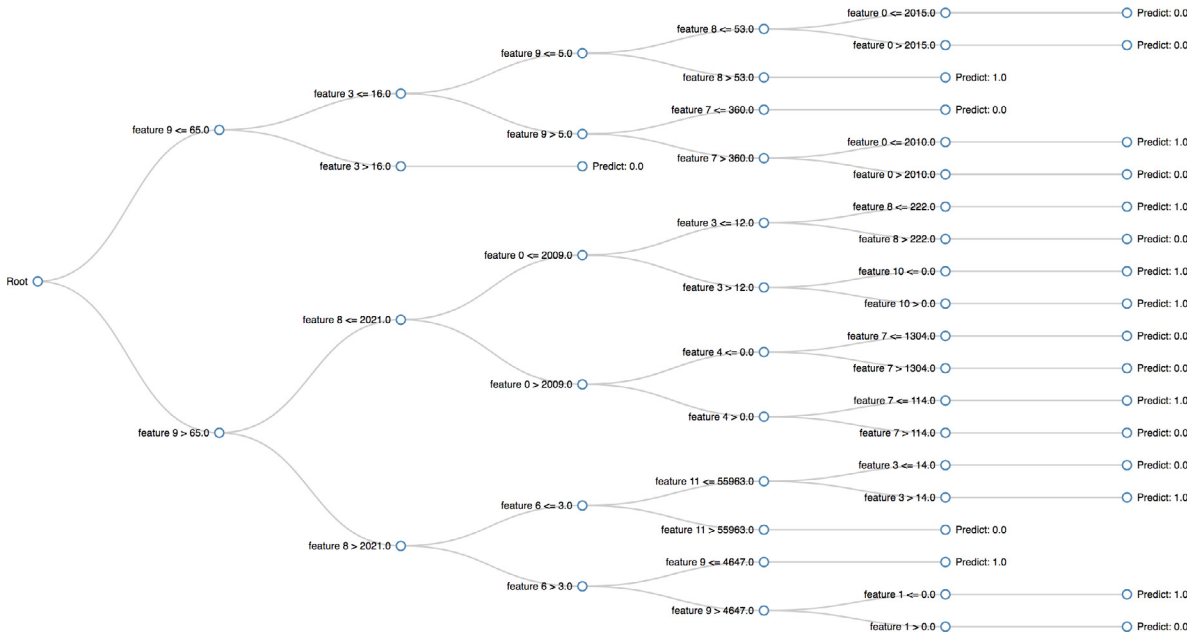

Fig. 4. Tree classifier for features for movieId 971380.

Based on our trained models and according to our features a classification model would be created. In Fig. 4 we can see the resulting classifier tree for the movie with movieId 971380.

We will train the models with our training data and we will do two parallel calculations: one with a single classification tree and another one with a random forest. After several execution of our predictions, we have used several metrics and the obtained results are shown in Tables 8–10.

In these tables are displayed the error values for 20 executions. For every execution, we calculate the following four metrics for the model built with *Classification Trees* (CT) as well as the model built with *Random Forests* (RF). In this way we can make a comparison between both models.

All these metrics are computed for every one of the different models (one per movie) and afterwards, the average of all of them will be calculated. That is the value that we show in every cell of the table.

We can observe that in general the accuracy error is slightly better for the random forest model as the one from classification tree. However in the average rating for the predictions, the classification tree takes the lead. That does not necessarily mean that the predictor is better than the one from the random forest, we have to bare in mind that we consider a movie suitable to be recommended if their rating is great or equal as 7; and average rating, for both models, is in general closer to 8 than to 7.

Thus we can say that random forest performs better than a single classification tree, which was something expected. As we can observe in Table 11 from the 20 executions we obtained an average accuracy error of 0.298 for the random forests

Table 8
Validation of predictions 1.

Iteration	1	2	3	4	5	6	7
avg rating (DT)	7.714	7.826	7.632	7.682	7.772	7.800	7.709
avg rating (RF)	7.279	7.295	7.534	7.475	7.695	7.525	7.244
avg accuracy error (DT)	0.356	0.348	0.335	0.327	0.336	0.324	0.327
avg accuracy error (RF)	0.318	0.306	0.301	0.296	0.295	0.283	0.298
avg RMSE error (DT)	0.588	0.583	0.573	0.564	0.570	0.561	0.564
avg RMSE error (RF)	0.555	0.544	0.542	0.535	0.531	0.520	0.538
avg f1 error (DT)	0.598	0.610	0.626	0.635	0.633	0.648	0.640
avg f1 error (RF)	0.601	0.613	0.624	0.623	0.637	0.650	0.631

Table 9
Validation of predictions 2.

Iteration	8	9	10	11	12	13	14
avg rating (DT)	7.805	7.894	7.768	7.674	7.758	7.694	7.697
avg rating (RF)	7.610	7.497	7.378	7.426	7.366	7.586	7.600
avg accuracy error (DT)	0.328	0.352	0.345	0.350	0.326	0.341	0.338
avg accuracy error (RF)	0.290	0.292	0.305	0.305	0.296	0.314	0.301
avg RMSE error (DT)	0.562	0.586	0.579	0.583	0.563	0.575	0.574
avg RMSE error (RF)	0.528	0.530	0.542	0.543	0.536	0.549	0.539
avg f1 error (DT)	0.628	0.615	0.613	0.605	0.625	0.617	0.628
avg f1 error (RF)	0.636	0.635	0.613	0.615	0.623	0.610	0.628

Table 10
Validation of predictions 3.

Iteration	15	16	17	18	19	20
avg rating (DT)	8.044	7.754	7.828	7.761	7.759	7.797
avg rating (RF)	7.651	7.408	7.396	7.667	7.507	7.475
avg accuracy error (DT)	0.313	0.354	0.337	0.318	0.361	0.334
avg accuracy error (RF)	0.274	0.315	0.292	0.287	0.304	0.292
avg RMSE error (DT)	0.550	0.587	0.573	0.556	0.595	0.569
avg RMSE error (RF)	0.512	0.552	0.532	0.525	0.544	0.532
avg f1 error (DT)	0.655	0.607	0.634	0.649	0.614	0.634
avg f1 error (RF)	0.663	0.606	0.645	0.653	0.632	0.634

Table 11
Accuracy error after 20 executions.

	Mean	Standard Deviation	Variance
Decision Trees	0.338	0.01325	0.0001757
Random Forest	0.298	0.01093	0.0001195

Table 12
Mean absolute Error comparison of models.

	MIPFGWC-CS	NHSM	FARAMS	HU-FCF	MJD	BSSB-RS
MAE	0.672	0.591	0.603	0.608	0.8	0.298

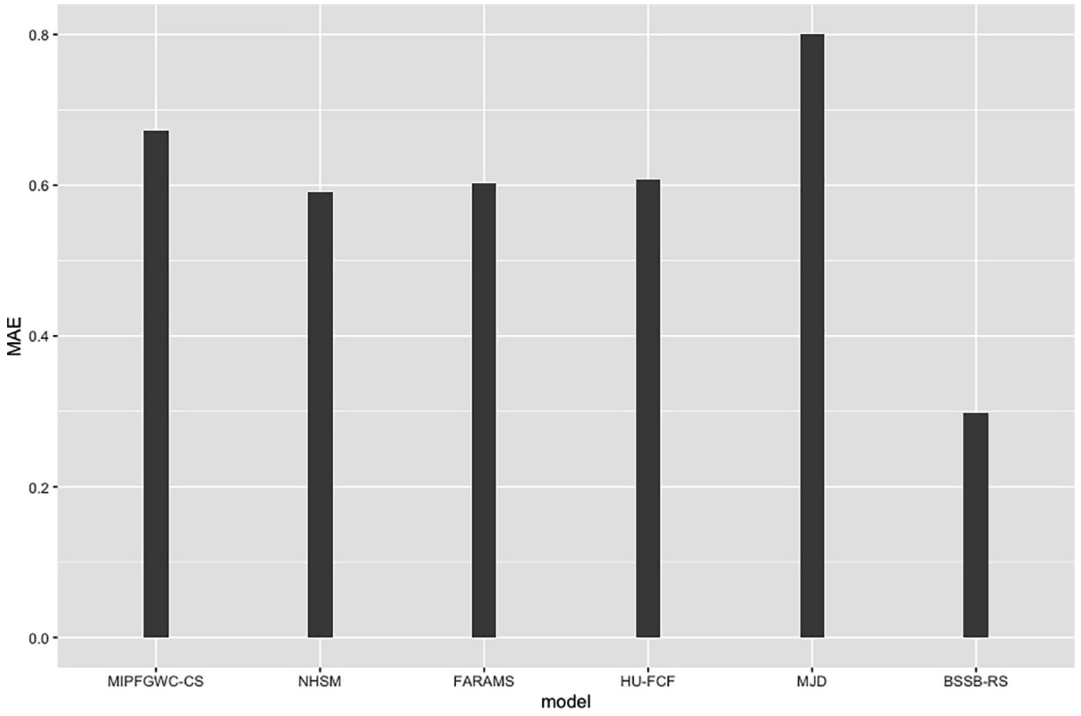

Fig. 5. RS Cold Start Model Comparison.

over the 0.338 for the single decision tree. Based on the standard deviation and variance we can also confirm that the obtained results are highly stable, especially in the case of the random forest.

Seeing these results we can assert that these results are very positive moreover taking into account the handicap that we do not use the previous rating data from the user to search for similar items. On the contrary the only data that we use from the user is not directly related with the items that we are going to create the recommendations for (movies in our case) but data from their social stream.

Furthermore, we compare the results of our Behavioural Social Stream Based Recommender System (BSSB-RS) with some other state-of-the-art works of new user cold-start problem [4,38].

It is important to remark that almost all the algorithms are not using any additional data for the decision making. They use instead some rating data from the users (they are not a purely zero ratings algorithm like ours). In addition to that the algorithms MIPFGWC–CS and HU-FCF are also using demographical data for their systems.

We can appreciate in the Table 12 and in Fig. 5 that our approach outperforms all new user cold-start proposed algorithm even though we are using an absolute zero ratings cold-start users.

It is important to remark that we can not compare these results with another non cold-start state of the art recommender systems since our model is only taking contextual data (Twitter stream data) as input. We only use the rating data of new users for building the model and for validation purposes. Therefore it would not be fair to compare our approach for a cold-start context with another context where a full rating history for all users is provided.

5. Conclusions and future work

We have proposed a recommendation approach based on a prediction model, using behavioural information extracted from social media to classify the users according to their behavioural profiles. Then, the users will not need to explicitly provide any personal information other than the source of their social media, helping in this way to alleviate the cold start problem. One of the main novelties of our system is that we obtained a rich and comprehensive data set that comprises two different data sources, for the rating data and for the social data, that are linked enabling the fulfilment of our experiments.

With the help of this implicit data obtained from the social media we can palliate the information gap that we have for new users of the system. Our algorithm is assisted with machine learning techniques, i.e., classification trees and random forest, which help us classify users assigning them a flag for every item indicating if it is suitable to be recommended or not. Although we have used classification trees and random forest, the most important idea of the approach is not the determined machine learning technique that is powering the algorithm but the integration of the behavioural data, obtained from social stream, and the rating data for creating recommendations.

The proposal has been validated in the movie recommendation environment, and the obtained results of the suggested predictions are truly satisfactory and therefore, the generated recommendations are in average very good since the results are outperforming other new user cold-start algorithms. Therefore we could assess that our algorithm (BSSB-RS) is an optimal asset in cold-start situations because it leverages the information we have in social media turning it into a very valuable data source enhancing the quality and precision in the decision making process and providing a much more accurate recommendation of items.

In this work we create our predictions establishing a direct relation between the user profile and the rating for a determined item. An eventual improvement to this process we could approach in future works would be establishing the relation between user profile and item's features (for example between Twitter profile and comedian genre or determined actor) increasing in this way the granularity of recommendations but also the complexity and execution time from algorithms. This could help us to obtain more granularity in the relations and therefore obtain better recommendations. We would have to deal with a significant more extensive amount of models that we would have to combine to obtain the final prediction. Furthermore, we have used some features for creating our model, however if we go deeper by doing some feature engineer and we extract more and more complex and relevant features from our social stream, i.e. performing sentiment analysis from the tweets the user have published, it could improve significantly the accuracy from the recommendations.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This paper has been developed with the FEDER financing of Project TIN2016-75850-R.

References

[1] J. Bernabé-Moreno, A. Tejeda-Lorente, C. Porcel, H. Fujita, E. Herrera-Viedma, Quantifying the emotional impact of events on locations with social media, Knowl.-Based Syst. 146 (2018) 44–57.
[2] J. Bernabé-Moreno, A. Tejeda-Lorente, C. Porcel, E. Herrera-Viedma, A new model to quantify the impact of a topic in a location over time with social media, Expert Syst. Appl. 42 (2015) 3381–3395.
[3] J. Bernabé-Moreno, A. Tejeda-Lorente, C. Porcel, E. Herrera-Viedma, Leveraging localized social media insigths for industry early warning systems, Int J Inform Technol Decision Making 17 (2018) 357–385.
[4] J. Bobadilla, F. Ortega, A. Hernando, J. Bernal, A collaborative filtering approach to mitigate the new user cold start problem, Knowl.-Based Syst. 26 (2012) 225–238.
[5] J. Bobadilla, F. Ortega, A. Hernando, A. Gutiérrez, Recommender systems survey, Knowl.-Based Syst. 46 (2013) 109–132.
[6] L. Breiman, Bagging predictors, Mach. Learn. 24 (1996) 123–140.
[7] L. Breiman, Random forests, Mach. Learn. 45 (2001) 5–32.
[8] L. Breiman, J. Friedman, C. Stone, R. Olshen, Classification and Regression Trees, Taylor & Francis, 1984.
[9] R. Burke, Hybrid web recommender systems, in: P. Brusilovsky, A. Kobsa, W. Nejdl (Eds.), The Adaptive Web, LNCS, 2007, pp. 377–408.
[10] R. Burke, A. Felfernig, M. Gker, Recommender systems: an overview, AI Magazine 32 (2011) 13–18.
[11] W. Carrer-Neto, M. Hernández-Alcaraz, R. Valencia-García, F. García-Sánchez, Social knowledge-based recommender system. application to the movies domain, Expert Syst. Appl. 39 (2012) 10990–11000.
[12] S. Charlotte-Ahrens, Recommender Systems: Relevance in the Consumer Purchasing Process, epubli, 2011..
[13] C. Chien, W. Yu-Hao, C. Meng-Chieh, S. Yu-Chun, An effective recommendation method for cold start new users using trust and distrust networks, Inf. Sci. 224 (2013) 19–36.
[14] A. Edmunds, A. Morris, The problem of information overload in business organizations: a review of the literature, Int. J. Inf. Manage. 20 (2000) 17–28.
[15] L. Esmaeili, S. Mardani, S. Golpayegani, Z. Madar, A novel tourism recommender system in the context of social commerce, Expert Syst. Appl. 149 (2020) 113301.
[16] F. García-Sánchez, R. Colomo-Palacios, R. Valencia-García, A social-semantic recommender system for advertisements, Inf. Process. Manage. 57 (2020) 102153.
[17] M. Goga, S. Kuyoro, N. Goga, A recommender for improving the student academic performance, Procedia – Social Behav. Sci. 180 (2015) 1481–1488.
[18] L. Gonzalez Camacho, S. Nice Alves-Souza, Social network data to alleviate cold-start in recommender system: a systematic review, Inf. Process. Manage. 54 (2018) 529–544.
[19] Grouplens, Movielens - movie recommendations. http://movielens.umn.edu/login..
[20] A. Hernando, B.J.F. Ortega, A. Gutiérrez, A probabilistic model for recommending to new cold-start non-registered users, Inf. Sci. 376 (2017) 216–232.
[21] T.K. Ho, Random decision forests, Proceedings of the 3rd International Conference on Document Analysis and Recognition, Montreal 1 (1995) 278–282..
[22] Y. Ho-Cho, J. Kyeong-Kim, S. Hie-Kim, A personalized recommender system based on web usage mining and decision tree induction, Expert Syst. Appl. 23 (2002) 329–342.
[23] L. Hoang-Son, Dealing with the new user cold-start problem in recommender systems: A comparative review, Inform. Syst. 58 (2016) 87–104.
[24] Y. Kim, K. Shim, Twilite: a recommendation system for twitter using a probabilistic model based on latent dirichlet allocation, Inform. Syst. 42 (2014) 59–77.
[25] B. Lika, K. Kolomvatsos, S. Hadjiefthymiades, Facing the cold start problem in recommender systems, Expert Syst. Appl. 41 (2014) 2065–2073.

[26] C. Martínez-Cruz, C. Porcel, J. Bernabé-Moreno, E. Herrera-Viedma, A model to represent users trust in recommender systems using ontologies and fuzzy linguistic modeling, Inf. Sci. 311 (2015) 102–118.
[27] H. Meng-Yen, W. Tien-Hsiung, L. Kuan-Ching, A keyword-aware recommender system using implicit feedback on hadoop, J. Parallel Distrib. Comput..
[28] S. Natarajan, S. Vairavasundaram, S. Natarajan, A. Gandomi, Resolving data sparsity and cold start problem in collaborative filtering recommender system using linked open data, Expert Syst. Appl. 149 (2020) 113248.
[29] M. Pazzani, A framework for collaborative, content-based and demographic filtering, Artif. Intell. Rev. 13 (5–6) (1999) 393–408.
[30] K. Pliakos, S. Joo, J. Park, F. Cornillie, C. Vens, W. Noortgat, Integrating machine learning into item response theory for addressing the cold start problem in adaptive learning systems, Computers Educ. 137 (2019) 91–103.
[31] I. Portugal, P. Alencar, D. Cowan, The use of machine learning algorithms in recommender systems: a systematic review, Expert Syst. Appl. 97 (2018) 205–227.
[32] J.R. Quinlan, Induction of decision trees, Mach. Learn. 1 (1986) 81–106.
[33] H.L. Reza Zafarani, Mohammad Ali Abbasi, Social Media Mining, Cambridge University Press, 2014.
[34] J. Rodger, An expert system gap analysis and empirical triangulation of individual differences, interventions, and information technology applications in alertness of railroad workers, Expert Syst. Appl. 144 (2020) 113081.
[35] A. Sahu, P. Dwivedia, V. Kant, Tags and item features as a bridge for cross-domain recommender systems, Procedia Computer Sci. 125 (2018) 624–631.
[36] B. Sarwar, G. Karypis, J. Konstan, J. Riedl, Analysis of recommendation algorithms for e-comerce, in: Proceedings of ACM E-Commerce 2000 conference, 2000, pp. 158–167.
[37] J. Serrano-Guerrero, E. Herrera-Viedma, J. Olivas, A. Cerezo, F. Romero, A google wave-based fuzzy recommender system to disseminate information in university digital libraries 2.0, Inf. Sci. 181 (2011) 1503–1516.
[38] L. Son, Dealing with the new user cold-start problem in recommender systems: A comparative review, Inform. Syst. 58 (2016) 87–104.
[39] L. Tao, J. Cao, F. Liu, Dynamic feature weighting based on user preference sensitivity for recommender systems, Knowl.-Based Syst. 149 (2018) 61–75.
[40] A. Tejeda-Lorente, C. Porcel, J. Bernabé-Moreno, E. Herrera-Viedma, Refore: a recommender system for researchers based on bibliometrics, Appl. Soft Comput. 30 (2015) 778–791.
[41] A. Tejeda-Lorente, C. Porcel, E. Peis, R. Sanz, E. Herrera-Viedma, A quality based recommender system to disseminate information in a university digital library, Inform. Sci. 261 (2014) 52–69.
[42] L. Terán, A. Oti-Mensah, A. Estorelli, A literature review for recommender systems techniques used in microblogs, Expert Syst. Appl. 103 (2018) 63–73.
[43] J.J. Thomas, P. Karagoz, A.B.B.P. Vasant, Deep learning techniques and optimization strategies in big data analytics, IGI Global (2019).
[44] I. Viktoratos, A. Tsadiras, N. Bassiliades, Combining community-based knowledge with association rule mining to alleviate the cold start problem in context-aware recommender systems, Expert Syst. Appl. 101 (2018) 78–90.
[45] J. Wei, J. He, K. Chen, Y. Zhou, Z. Tang Collaborative filtering and deep learning based recommendation system for cold start items, Expert Syst. Appl. 69 (2017) 29–39.
[46] H. Wu, K. Yue, Y. Pei, B. Li, Y. Zhao, F. Dong, Collaborative topic regression with social trust ensemble for recommendation in social media systems, Knowl.-Based Syst. 97 (2016) 111–122.
[47] X. Yang, Y. Guo, Y. Liu, H. Steck, A survey of collaborative filtering based social recommender systems, Comput. Commun. 41 (2014) 1–10.
[48] Y. Zhang, Z. Shi, W. Zuo, L. Yue, X. Li, oint personalized markov chains with social network embedding for cold-start recommendation, Neurocomputing Available online December 2019..
[49] S. Zhoubao, H. Lixin, H. Wenliang, W. Xueting, Z. Xiaoqin, W. Min, Y. Hong, Recommender systems based on social networks, J. Syst. Softw. 99 (2015) 109–119.