Coping with the Persistent Cold-start Problem

Siarhei Bykau
University of Trento
bykau@disi.unitn.eu

Georgia Koutrika
HP Labs
koutrika@hp.com

Yannis Velegrakis
University of Trento
velgias@disi.unitn.eu

ABSTRACT

Recommender systems predict items a user is likely to like using historical data about users and items. A key challenge is how to provide recommendations when historical data is sparse or missing, known as the cold-start problem. Current solutions to this problem assume that given an item and a user, the recommendation process misses historical data only about one of them but not both. In this paper, we are interested in the challenging more severe form of the cold-start problem of new-user/new-item. In particular, we are interested in cases where a system collects historical data about users and items but produces recommendations mostly for new or anonymous users and about new or evolved items. We present methods that can be used to deal with this problem, study them and present our experimental findings.

1. INTRODUCTION

Recommender systems help users select items of interest by predicting items that the user will most likely like. There are two main categories of recommender systems: the *content-based* systems that exploit selections that a user has made in the past [13] and the *collaborative filtering* systems that recommend items selected by users with similar preferences [20]. Some times, there is no previous information about a user or an item in order to allow a recommendation to be made. This is known as the *cold-start problem*. Depending on whether the missing information is about a user's past choices, an item's past selections by users, or both, the cold start problem is found under the names *new-user*, *new-item* or *new-user/new-item*, respectively.

Solutions to the cold start problem vary, but are mainly based on the idea of using other sources of information in order to make a recommendation. For instance, one way is to assume that the most popular item in general will also be popular to a user for which we have no history. An initial interview [17] or a social network [6] can provide a first idea of the user preferences and guide the selection of items to recommend. For new items, ontologies or other forms of external information can provide a better picture of the nature of the object and guide the selection of the user to which the item is to be recommended [12]. As an alternative, the features of users and items can be used (if this is possible) to capture the similarities between users and items and then recommend each item to the users to which it is most similar [1]. Finally, since no solution works perfect all the time, hybrid approaches may be followed [16].

Most of the existing works on the cold start problem focus on the new-user or the new-item situation. However, there are many practical scenarios in which there is a continuous lack of historical information both for items and users. Examples include cases in which the items have a short life span, or lose very fast their value, while at the same time there is a continuous influx of new users, practically, the user population follows the power law distribution with respect to participation frequency, with most users lying on the long-tail part. One such application are news sites like CNN or NYTimes, where new or unknown users appear constantly in the system with the majority being interested in new content [11]. At the same time, an article can become important and get recommended only when it receives a significant amount of visitors, but by the time this happens, its content has become "old" news, and the article is by default less important. Another example are course recommendations. In such system, the users are either freshmen (i.e., without history) or have too short history (max 4/5 years and on completely diverse courses) [14] to allow for a good user profiling. At the same time, the characteristics of the courses are changing frequently in terms of content, instructor, exam, teaching methodology, etc, to a point that every new offering of a course can be considered as a new course. Note, considering the information about curriculum requirements and course prerequisites can be used in this context but it is not always available and therefore in this work we assume that no external source of information is given.

Dealing with applications like those just mentioned where the the new-user/new-item cold start problem is permanently present is a challenging task. Neither collaborative filtering, nor content-based solutions can be applied since there is no historical information about the new users or the new items, respectively. Randomly making some suggestions until a significant amount of information has been collected and informative recommendations can be made, a solution to which many applications often resort [15], will not work either due to the continuous appearance of new items and users. To the best of our knowledge, no other work has studied this problem per se, but have all considered it as a small side-problem of a recommendation system. We believe that the applications in which this type of cold-start problem is present are too many and important to ignore. Our focus in this work is exactly on this.

We study three different methods for dealing with the problem. Each method leverages on the item features to produce recommendations. The first method, that we refer to as the *Similarity-based*, sees each item as a whole and is based on the assumption that the

more similar a new item is to an already popular item, the more popular the new item would also be. The second method, referred to as the *Feature-based*, sees the items as sets of independent features. It measures the popularity of each feature individually, and then generates a popularity rating based on the collective popularity of the individual features. For instance, if courses by professor 'Ullman' have been popular in the past, then new courses taught by 'Ullman' are very likely to be popular courses for new students. The third method, referred to as *Maximum Entropy-based*, takes into consideration popularity of combinations of features. By using the maximum entropy principle [9] it predicts the popularity of the combination of features in new items. For instance, the method may identify that courses taught by 'Ullman' are popular only when they are related to data management, thus, a course on VLSI taught by Ullman will not be highly recommended. We evaluate the performance of these methods on the new-user/new-item cold-start problem based on: (i) the correctness of ranking, using the normalized Discounted Cumulative Gain (nDCG), (ii) the prediction accuracy, using the Root Mean Square Error (RMSE). We also compare them to an adapted version of the Pairwise preference regression method [15] that is closest to our methods, also dealing with the new-user/new-item cold start problem. Since we do not assume the existence of any auxiliary sources, e.g., social data, we cannot compare to other approaches that operate on such assumptions.

2. MOTIVATION

Consider a system used by students in a university for evaluating courses that they have taken. Figure 1 illustrates a portion of the information kept in the system. It is Sept 2013 and a new student has just arrived and is wondering whether to take the course `cs421` offered in the 3rd trimester by professor Fox in the area of DB, with an oral final exam. This combination of professor, type of exam, area, etc., has not been seen before, so it is not clear whether this course would be liked by the students or not. Furthermore, since the student is new, there is no knowledge about her preferences.

One can look at Figure 1 and observe that the courses `cs123` and `cs124`, which differ substantially from the `cs421`, have received very low evaluations, while the course offering of `cs343` in 2011 that, similarly to the new course `cs421`, is also in DB, taught by Fox and in the 3rd trimester (with a different type of exam, though), has received an avg evaluation. This means that very likely `cs421` will also receive an avg evaluation if taken.

Another way to reason is by looking at the characteristics of the course individually. In particularly, it can be seen that in average, when an oral examination is in place, the course receives an above avg evaluation. The same applies for the DB topic, and to some degree with the 3rd semester. Thus, even if courses with oral examination are typically rated low or avg, the new offering of `cs421` will be more likely to receive an avg or high evaluation.

Nevertheless, one can see that of the three times that Fox taught a course with an oral exam, the course received a high evaluation. The same holds when an oral examination is done in the 3rd trimester, while whenever Fox taught DB, the satisfaction was not very high. As a result, `cs421` will most likely satisfy the student and should be recommended to her.

3. RELATED WORK

Content-based filtering systems recommend items based on profiles built on past user choices [13] while collaborative filtering systems recommend items chosen by like-minded users [20]. When little information about users and items exists, these recommendation methods underperform. Linear factor models can be used to smooth out noise [2]. However, these methods cannot be applied to the cold start problem [18], i.e., when no historical information about an item and a user exists.

cid	year	area	instructor	trimester	exam	student	rating
cs343	2011	DB	Fox	1	written	s5	*avg*
cs343	2010	DB	Fox	1	written	s6	*low*
cs343	2011	DB	Fox	3	written	s7	*avg*
cs241	2010	PL	Smith	2	oral	s5	*avg*
cs241	2010	PL	Smith	2	oral	s9	*low*
cs241	2011	PL	Smith	2	oral	s5	*high*
cs241	2011	PL	Smith	2	oral	s1	*avg*
cs241	2010	PL	Smith	2	oral	s2	*low*
cs241	2011	PL	Smith	2	oral	s4	*high*
cs120	2008	OS	Fox	1	oral	s4	*low*
cs120	2009	OS	Fox	1	oral	s10	*high*
cs120	2010	OS	Fox	1	oral	s11	*high*
cs120	2011	OS	Fox	1	oral	s15	*high*
cs100	2011	HW	Bell	3	oral	s2	*high*
cs100	2011	HW	Bell	3	oral	s3	*high*
cs100	2011	HW	Bell	3	oral	s4	*avg*
cs100	2010	HW	Bell	3	oral	s5	*high*
cs100	2010	HW	Bell	3	oral	s6	*high*
cs400	2011	DB	Newton	3	oral	s10	*high*
cs400	2010	DB	Newton	3	oral	s20	*high*
cs400	2011	DB	Newton	3	oral	s18	*high*
cs123	2011	TH	Megan	2	project	s6	*low*
cs124	2011	GR	Megan	4	project	s12	*low*
cs123	2010	TH	Thomson	4	project	s14	*low*

Figure 1: A Course Evaluation System Data

[New-Items/Existing-Users] For making recommendations on new items, external information has often been used like social tags [5] or ontologies [12] to provide a better understanding of the items and the popularity they may have among the users.

[Existing items / New users] Often, in a recommender system, new users comprise the vast majority. Approaches to elicit new user preferences range from interviews [17] to exploitation of social and trust networks [6]. Probabilistic models [19] have been used to combine user and item meta data to make recommendations. For the same purpose, a regression-based latent factor model [1] has also been proposed. Hybrid approaches assemble several independent models [16] whereas other frameworks learn user-item affinity and then use it for predictions [15].

[New items / New users] Finding recommendations for new users on new items is clearly the hardest form of cold-start problem. For that, a regression model for user/item pairs that leverages item features, user ratings and demographics has recently been proposed [15]. This work is very close to ours, but it still uses some external information. Instead, our work focuses on the clean case of cold start where no other external information is assumed, neither for the new items, nor for the new users.

4. PREDICTING EVALUATIONS

Assume the existence of an infinite set $\mathcal{U}$ of users, $\mathcal{F}$ of features and a domain $\mathcal{D}$ of values. An item i is a tuple of features $\langle f_1, \ldots, f_n \rangle$. Let $\mathcal{I}$ represent the set of all possible items. An *evaluation* is a tuple $\langle i, u, v \rangle$ that denotes the assignment of a value $v \in \mathcal{D}$, referred to as rating, to an item i from a user u. Often, we will use the notation $r_{u,i}$ to denote the value v, and F_i to denote the set of features $\{f_1, f_2, \ldots, f_n\}$ of the item i.

Given a finite set of evaluations $R \subset \mathcal{I} \times \mathcal{U} \times \mathcal{D}$, a new item i_o is essentially an item for which there is no evaluation in R. Similarly,

a new user u_o is a user who has provided no evaluations in R.

NEW-USER/NEW-ITEM RECOMMENDATION PROBLEM. Given a set of evaluations $R \subset \mathcal{I} \times \mathcal{U} \times \mathcal{D}$, a set $\mathcal{I}_o$ of new items, and a new user u_o, our goal is to: (i) compute a rating r_{i_o} for every item $i_o \in \mathcal{I}_o$ according to the evaluation each one is expected to receive if taken by user u_o, and (ii) produce a ranking of the items in $\mathcal{I}_o$ using their computed ratings.

4.1 Linear Regression

As a baseline approach we use the Pairwise Preference Regression algorithm [15] which is the closest state-of-the-art solution which can address the new-user/new-item problem in an environment similar to ours. However, we cannot directly apply the method since it requires the users to have some features (e.g. user demographics, age information). To adapt the algorithm, we assume that all users have only a single common feature. Under this assumption, i.e., that the user feature vector has a single feature, we reduce the Pairwise Preference Regression algorithm to the *linear regression* problem, i.e., every rating is a weighted sum of the features of the corresponding item.

4.2 Similarity-based Recommendation

Given a new item i_o and a new user u_o, one approach to predicting how much u_o would like i_o is to see how other users have rated similar items in the past. The more similar i_o is to popular past items, the more likely it is that i_o will be a popular item itself, and hence the more likely it is that i_o will be liked by u_o, i.e., the higher its recommendation score should be.

In order to measure the similarity $sim(i, i_o)$ between two items, i and i_o, different similarity functions can be used, but a prevalent one is the Jaccard similarity [7] that measures how similar two sets are. Then, the expected rating of the new item i_o can be computed as the average of the existing ratings, weighted according to the similarity of the respective items with i_o. Specifically,

$$r_{i_o} = \frac{\sum_{\langle i,u,r\rangle \in R} sim(i, i_o) r}{\sum_{\langle i,u,r\rangle \in R} sim(i, i_o)}$$

The formula makes sure that the ratings of the items that are more similar to i_o contribute more to the final rating for i_o. Considering all the existing ratings in R may not only be computationally expensive, it may also create a lot of noise since many items with low similarity will be actively affecting the results. To avoid this issue we consider only the ratings in R of the top k nearest neighbors [3].

4.3 Feature-based Recommendation

The Similarity-based Recommendation considers each item as a monolithic structure on which each user rating is referring to. However, an item consists of a number of features, and a rating assigned to an item as a whole is actually a reflection of how the individual features of the item are evaluated. Hence, if courses with professor Fox have been rated highly in the past, then a new course by professor Fox will probably be a good recommendation, since professor Fox has been a 'success ingredient' in the past.

With this in mind, one can compute a rating for an individual feature based on the ratings of the items with this particular feature. In our case we compute an average of ratings of all items which have the feature. Then, a rating prediction for a new item can be computed by combining the ratings of its individual features. This approach assumes that features are independent.

Assuming that a rating of an item translates equally to ratings of its features, the received rating $rt(f)$ of an individual feature f is the average rating of all the items that have that feature, i.e.,

$$rt(f) = \frac{\sum_{\langle i,u,r\rangle \in R^f} r}{|R^f|}$$

with R^f denoting the set of evaluations on items that have the feature f, i.e., $R^f = \{\langle i,u,r\rangle |\ \langle i,u,r\rangle \in R \wedge f \in F_i\}$. Note, treating features independently and use the average to make a prediction may lead to the feature rating interference (e.g. a high rated feature John appears with many average rated features making the prediction average). We overcome this limitation in the max entropy method (see Section 4.4).

4.4 Max Entropy-based Recommendation

The Similarity-based and the Feature-based Recommendation approaches represent two extremes on how to interpret existing user evaluations. The former assumes that the evaluation is for the maximum number of features that are common with the item for which a prediction needs to be made, while the latter assumes that it is for each feature independently. The former fails to acknowledge the fact that the evaluation may be for fewer features, the latter that it may be for more than one and recognize the joint relationships between features. An approach that lies somewhere between the two is to associate the rating to every possible combination of features that also exist in the new item for which the prediction needs to be made.

Of course considering all the possible combinations of features is computationally expensive due to their exponential number. Furthermore, it has the risk of not weighting well the various alternatives. Ideally, we want to recognize *preference patterns*, i.e., interesting combinations of features, that capture user preferences as expressed in the existing evaluations made by users on different items. For this purpose, a more principled approach is to use the notion of max entropy [9] which tries to maximize the amount of information that the features and any combination of them are providing.

Intuitively, the idea is to treat the (non-)existence of a feature and the rating that an item receives as random variables, and find a joint probability distribution that satisfies the existing evaluations and introduces the minimum possible additional information, i.e., that is the closest to the distribution with the maximum entropy.

More specifically, given n features, $f_1, \ldots, f_n$, we define $n{+}1$ random variables x_i, with $i{=}1..(n{+}1)$. The variable x_i (for $i{=}1..n$) corresponds to the feature f_i and gets values 1 or 0 meaning that the feature f_i is present or not. The n-(th+1) variable, denoted as r, can take any of the values that can be provided by a user as an evaluation to an item. Having these n+1 random variables, their joint distribution, denoted as $p(r, \vec{x})$ can be defined, where $\vec{x}$ is a vector of the random variables $x_1 \ldots x_n$. Each of the evaluations that previous items have received can be represented as a vector $\vec{c}$ of n binary values, indicating the existence or not of the respective features and a value v representing the evaluation received. The vector $\vec{c}$ and the value v can be interpreted as an observation of the $n+1$ random variables.

PREFERENCE PATTERN. We define a preference pattern as a set of features and a rating. Given the set of n features of the data, and the domain of the ratings, we can create a list $\mathcal{P}$ of all the patterns.

Viewing the existing evaluations as observations of the $n+1$ variables, the goal is to find the joint probability distribution $p(r, \vec{x})$ that satisfies the constraints set by the observations, i.e., is consistent with them, and at the same time is as close as possible to the probability distribution $\pi(r, \vec{x})$ that has the maximum entropy. To do the former, for each pattern q_j in $\mathcal{P}$, with $j{=}1..|\mathcal{P}|$, a rule is

created of the form:

$$\sum_{r,\vec{x}} p(r,\vec{x})k(r,\vec{x} \mid q_j) = m_j/m \quad (1)$$

where $k(r,\vec{x} \mid q_j)$ is a function that returns 1 if r and $\vec{x}$ satisfies the pattern q_j and 0 if not. Satisfying the pattern q_j means that the features of the vector $\vec{x}$ and the variable r contain those of the pattern. The number m_j is the number of evaluations in the dataset that satisfy that pattern. The number m is the total number of evaluations in the dataset. The rules that are generated in practice do not really have to be for all the possible patterns but only for those for which m_j is above a specific cut-off value.

EXAMPLE 4.1. *Assume a pattern q' that has the features (PL, $Smith$) and the evaluation $high$. In the dataset of Figure 1, out of the total 24 evaluations, there are two that contain these two features and have the rating $high$. As such, we require from the joint distribution function we are looking to find, to return the value $\frac{2}{24}$ when its arguments are the vector $\vec{x}$=(PL, $Smith$) and the value r=$high$, by creating the rule*

$$\sum_{r,\vec{x}} p(r,\vec{x})k(r,\vec{x} \mid q') = 2/24$$

The probability distribution $p(r,\vec{x})$ is of the following form [4]:

$$p(r,\vec{x}) = \pi(r,\vec{x}) \prod_{j=0}^{|\mathcal{P}|} \lambda_j^{k(r,\vec{x}|q_j)} \quad (2)$$

Thus, in order to find the probability distribution $p(r,\vec{x})$ we need to find such $\lambda_0,\ldots,\lambda_m$ that every constraint of the form

$$\sum_{r,\vec{x}} \left(\pi(r,\vec{x}) \prod_{h=0}^{|\mathcal{P}|} \lambda_j^{k(r,\vec{x}|q_h)}\right) k(r,\vec{x} \mid q_j) = m_j/m \quad (3)$$

is satisfied. To achieve this, we utilize the method of Generalized Iterative Scaling algorithm (GIS) [4]. GIS can be applied only if the following condition is true:

$$\forall r,\vec{x} \sum_j k(r,\vec{x} \mid q_j) = const \quad (4)$$

and in order to guarantee this we add one additional constraint:

$$k(r,\vec{x} \mid 0) = d_{max} - \sum_{j=1}^{|\mathcal{P}|} k(r,\vec{x} \mid q_j) \quad (5)$$

where $d_{max} = max\{k(r,\vec{x} \mid q_j) \mid j = 1..|\mathcal{P}|\}$ The above two conditions normalize the preference patterns and ensure that for every pattern there is at least one non zero feature. Under these constraints GIS converges to an optimal solution (see the details in [4]).

The idea of the GIS algorithm is to start with an approximation of the $p(r,\vec{x})$ function that uses some initial values of the parameters λ_j, and then iteratively update these parameters generating a better approximation and heading towards the satisfaction of all the constraints set by the existing evaluations in the dataset. At the moment all the constraints are found to be satisfied, the process stops and the values of the λ_js are used to generate the solution. In every iteration, the values of the λ_js (denoted as $\lambda_{j,t}$) are updated to the new values $\lambda_{j,t+1}$ according to the following formula:

$$\lambda_{j,t+1} = \lambda_{j,t}\Big(\frac{m_j/m}{S_{j,t}}\Big)^{\frac{1}{d_{max}}} \quad (6)$$

where $p_t(r,\vec{x})$ is an estimation of $p(r,\vec{x})$ at iteration t and

$$S_{j,t} = \sum_{r,\vec{x}} p_t(r,\vec{x})k(r,\vec{x} \mid q_j)$$

To compute a prediction for an item with a set of features $\vec{c}$, we need to make a prediction for the vector of features $\vec{c}$. For this we use the conditional probability of ratings with known set of features, i.e. $p(r \mid \vec{x} = \vec{c})$, and consider as prediction the mean of that distribution:

$$r_{i_o} = r_{\vec{c}} = mean_r(p(r \mid \vec{x} = \vec{c}))$$

5. EXPERIMENTAL EVALUATION

To evaluate our solutions we have conducted a number of experiments on two real datasets on a 4GHz machine with 4G of memory. The techniques were all implemented in Java. The Maximum-entropy approach was adapted accordingly to our requirements from the Apache OpenNLP project[1].

5.1 Metrics

We quantitatively measured the effectiveness of our proposals from two aspects: *predictability* and *coverage*. The former aims at measuring the accuracy of our recommendations with respect to the available data. The latter measures the number of items and users for which we can successfully produce a recommendation.

For the *predictability* we use two metrics. We use the *Root Mean Square Error (RMSE)* for measuring the effectiveness of the individual rating predictions, i.e., how well we can predict a rating for a new item. The higher the RMSE is, the further the predicted ratings are from the actual ratings.

The second metric measures the effectiveness of the order of the recommended items based on their predicted rating. This metric uses the Normalized Discounted Cumulative Gain (NDCG) [8], which intuitively measures how a list differs from another based on the assumptions that (i) highly relevant items are more useful when appearing earlier and that (ii) highly relevant items are more useful than marginally relevant items.

For the *coverage*, we simply need to measure the ratio between the number of items for which the algorithm is able to produce recommendations and the total number of items.

5.2 Datasets

Course Rating. As our first dataset, we used a dataset from the Stanford course evaluation system [14] containing ratings across different departments between the years 1997 and 2008. In the experiments we used the ratings from the CS department where we had 9799 course ratings for 675 individual courses with 193 instructors in 17 terms within the period mentioned above. For every course we also have a number of features like the course title, description, type of exam, etc.

To measure the accuracy and the coverage we consider a part of the data as the given and another part for testing. A pair of given-test data is created as follows. We consider a specific point of time t which coincides with the end of some specific term. We consider any evaluation that has been made before that time t as given (i.e., part of the training data), and the evaluation of the 3 terms after time t as the ground truth (i.e., test data). This partition scheme reflects very accurately the reality since the only information that the users taking and evaluating these courses in the 3 terms after time t have the evaluations that have been received before time t, and the courses in these 3 terms have never been evaluated before, so we are actually in a real cold-start situation. The reason we can safely consider 3 terms after time t (and not only one) is that courses are very rarely offered more than once in the same academic year, thus, all the offerings in these 3 terms are unique. At the same time, using 3 terms gives the opportunity to have more test data. Since we

[1]http://opennlp.apache.org/

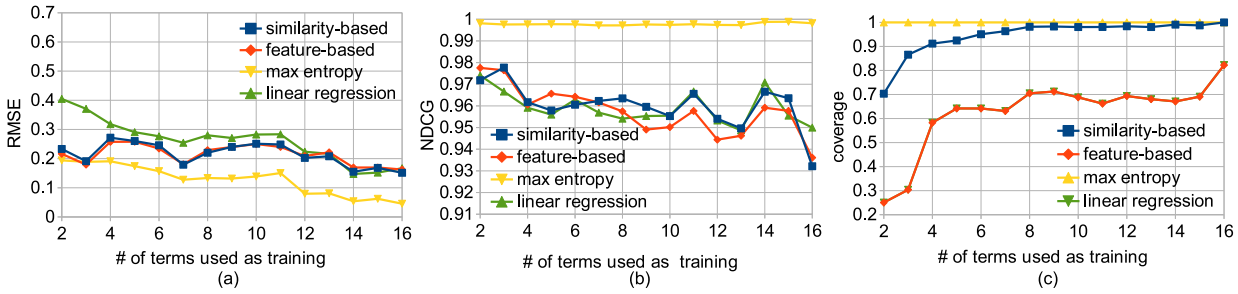

Figure 2: Accuracy and coverage for different numbers of terms (n_{train}) used as training on the Course Rating dataset

wanted to investigate the version of the cold-start problem in which we had new users and new items, we are also removing from the testing set all the evaluations made by users that have also evaluated some other course in the given data set.

Note that every course offering is by default a new item. Even if the same course has been offered in previous years, since we operate at the offering level, two offerings, even if they agree in all features but the year, are considered different items. If the same course re-appears, this is taken into account because its features contribute several times towards its rating.

Furthermore, for a given point of time t we have decided not to consider all the evaluations that have been made in terms before time t because this will include very old evaluations that may be of small use for predicting evaluations for new items. As such, we consider only the n_{train} terms that immediately precede time t, and we have run experiments for different values of this number.

MovieLens. The second dataset we have used is from the MovieLens[2] web site and consists of 100K ratings made by 1000 users for around 1700 movies with 42000 unique features and on average 39 features per movie. As features we used characteristics like title, year, genre, etc. For each movie we extracted from IMDB the director and the actors and used them as features, which explains the many features that we have for the movies. We partition the data into 5 equal datasets and then use one dataset for testing and the rest for training, giving a ratio of testing to training data 1:4, and we repeat the experiment for each of the partitions.

5.3 Experiment Planning and Tuning

We compare our proposed approaches, i.e., the Similarity-based approach, referred to as `similarity-based`, which uses the k most similar items in the training set to make a prediction for the expected rating of a new item, the Feature-based approach (`feature-based`) that assumes the independence among the features, and the maximum entropy approach (`max entropy`) that exploits the maximum entropy principle for generating predictions, plus the linear regression (`linear regression`) which is adapted from the Pairwise Preference Regression [15]. Also we experimented with the Naive Bayes classifier which showed a poor performance with respect to other methods and therefore was omitted for the sake of space.

Since the `similarity-based` and the `feature-based` approaches may not produce recommendations for every item in the testing data (their coverage is less than 1 as we will see), we are reporting the RMSE and NDCG values only for the items that both approaches can predict a (non-zero) rating. For `similarity-based`, we ran a number of experiments on the Course Ratings dataset where we varied the value of k from 1 to 17, measuring at the same time the RMSE, NDCG and coverage. As a result, we found that k with the value 5 leads to the highest recommendation accuracy both for the RMSE and NDCG metrics.

For `max entropy`, we use the shrinkage parameter[15] whose intuition is as follows. The actual average rating of any item is by default the average of all the ratings of all the items that exist in the database. However, the more evidence, i.e., evaluations, there exists for an item, the more likely it is that its actual value is shifted towards the average that these evaluations generate. In the experimental evaluation we use the shrinkage parameter of 2 which leads to the highest accuracy of `max entropy` on the Course rating dataset.

5.4 Effectiveness & Efficiency

Effect of the number of existing evaluations. To investigate the importance of the number of available evaluations in the recommendation process, we run our algorithms multiple times, while we gradually increased the number of terms used as training from 2 to 16. Figure 2 shows the results for each experiment, where chart (a) shows the individual rating predictability (RMSE), (b) the effectiveness of course ordering (NDCG) and (c) the coverage.

We observe that RMSE is improving with the growing number of terms used for training for the `similarity-based`, `feature-based` and `max entropy` methods, because the more information they have on features the better their predictions are. All three algorithms outperform the adapted version of the method in [15] (`linear regression`) for the individual rating prediction accuracy. (Recall that the higher the RMSE is, the further the produced results are from the ground truth.)

On the other hand, the NDCG (Figure 2(b)) remains almost stable which means that increasing the number of ratings in the training dataset doesn't lead to much improvement in the predictability of the right order in which the items are recommended. The interesting observation is that `max entropy` always gives a high NDCG value, which means that even if the method fails to predict the exact evaluation that a new item will receive from the users, the relative values of the evaluations it predicts are similar to the relative values the items will actually receive from the users, leading to the same ordered list of recommendations.

Regarding coverage (Figure 2(c)), the `similarity-based` and `feature-based` improve the coverage with the number of training terms, although the `feature-based` method always has a lower value. The latter is due to the fact that if an item has even one feature that does not appear in any evaluation used for training, the approach will make no prediction. In contrast, the `similarity-based` approach will not be able to make a prediction for an item only if all the item features are missing from the items in the training set, which is less likely to happen. `max entropy` always has a coverage of 1 due do the nature of algorithm (see Section 4.4).

Effect of the number of features. To study the effect of the number of features on the accuracy and coverage we use the MovieLens

[2]http://www.grouplens.org/node/73/

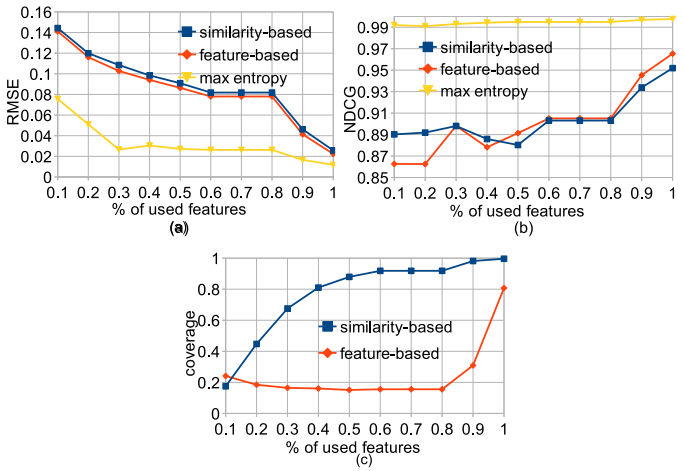

Figure 3: Effect of the number of features in the various approaches.

dataset (that has a larger number of features than the Course Rating dataset). We derive sets of new items from the original dataset by omitting from the items a percentage of their features. We do this for values from 10% to 100% (100% means that all features are considered) with the step of 10%. For each dataset created, we conducted a 5-fold cross-validation and recorded the average values for RMSE, NDCG and coverage shown in Figure 3. Note that we do not show the results of the `linear regression` algorithm in the following experiments since the solution proposed in [15] fails for a large matrix of features (since the operation of finding the inverse matrix is computationally expensive) and thus we could use it only for the course dataset.

Figure 3(a) shows that the predictability of individual ratings (RMSE) improves with the number of features, and all algorithms illustrate a similar behavior, with `max entropy` performing the best.

In contrast to the experiment on the training dataset size, here NDCG is improving with a growing number of features since more information is becoming available for making the right predictions (Figure 3(b)). `max entropy` is again showing a very robust behavior. Even with few features, its success rate is very high. It is able to effectively compose information provided by ratings on items with overlapping features and make the right predictions.

The coverage dynamics of `similarity`- and `feature-based` are different (Figure 3(c)). The former is increasing fast but the rate of increase is getting smaller as the number of features gets larger, since the algorithm starts having enough features to make the predictions, and some more new features do not actually add new items. Asymptotically, the coverage grows to the full coverage. The `feature-based` coverage remains for many different numbers of features around 0.2 and only when the algorithm starts considering 80% of all the available features the coverage increases sharply to 0.8.

Efficiency. The efficiency of the proposed algorithms is high and close to run-time for the Course ratings and MovieLens datasets. For `k-neighbor` we obtain recommendation in less than a sec having all the data in main memory. For `global feature` we first compute the rating of every feature available in the testing data and then compose them to make recommendations. This also leads to the recommendation generation in almost a run-time fashion. For the `maxent` we use the approximation which is explained in [10] has the complexity of $O(NPA)$ where N is the size of training dataset, P is the number of predictions and A is the number of features in an item we are making a prediction to.

6. CONCLUSION

We have identified various application scenarios in which the cold start situation is permanently present, thus, traditional solutions that call for a learning period at the beginning are inapplicable. We have investigated different solutions, one based on item similarity, one on the independence of features and one on max entropy for the new-user/new-item cold-start problem with the assumption that there is absolutely no form of external auxiliary information. Finally, we have evaluated the effectiveness of these methods, not only between them but also compared to other related approaches. As the future work, we plan to investigate how temporal dynamics of ratings can be used to improve the accuracy as well as to make use of the data from external sources (e.g. social, user surveys).

References

[1] D. Agarwal and B.-C. Chen. fLDA: matrix factoriz. through latent dirichlet allocation. In *WSDM*, pages 91–100, 2010.

[2] D. Billsus and M. J. Pazzani. Learning collaborative information filters. In *ICML*, pages 46–54, 1998.

[3] T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Trans. Inform. Theory*, pages 21–27, 1967.

[4] J. N. Darroch and D. Ratcliff. Generalized iterative scaling for log-linear models. *The Annals of Mathematical Statistics*, 43(5), 1972.

[5] D. Eck, P. Lamere, T. Bertin-Mahieux, and S. Green. Automatic generation of social tags for music recommendation. In *NIPS*, 2007.

[6] I. Guy, N. Zwerdling, D. Carmel, I. Ronen, E. Uziel, S. Yogev, and S. Ofek-Koifman. Personalized recommendation of social software items based on social relations. In *RecSys*, pages 53–60, 2009.

[7] P. Jaccard. Distribution de la flore alpine dans le bassin des Dranses et dans quelques régions voisines. *Bulletin de la Société Vaudoise des Sciences Naturelles*, 37:241–272, 1901.

[8] K. Järvelin and J. Kekäläinen. Ir eval. methods for retrieving highly relevant documents. In *SIGIR*, pages 41–48, 2000.

[9] E. T. Jaynes. Information theory and statistical mechanics. *Physical Review Series II*, pages 620–630, 1957.

[10] R. Lau, R. Rosenfeld, and S. Roukos. Adaptive language modeling using the maximum entropy principle. In *HLT*, pages 108–113, 1993.

[11] J. Liu, P. Dolan, and E. R. Pedersen. Personalized news recommendation based on click behavior. In *IUI*, pages 31–40, 2010.

[12] S. E. Middleton, H. Alani, and D. C. D. Roure. Exploiting synergy between ontologies and recommender systems. In *WWW*, 2002.

[13] D. Mladenic. Text-learning and related intelligent agents: A survey. *IEEE Intelligent Systems*, 14(4):44–54, July 1999.

[14] A. G. Parameswaran, G. Koutrika, B. Bercovitz, and H. Garcia-Molina. Recsplorer: recommendation algorithms based on precedence mining. In *SIGMOD'10*, pages 87–98, 2010.

[15] S.-T. Park and W. Chu. Pairwise preference regression for cold-start recommendation. In *RecSys*, pages 21–28, 2009.

[16] M. Pazzani. A framework for collaborative content-based and demographic filtering. *AI Review*, 13(25-6):393–408, 1999.

[17] A. M. Rashid, I. Albert, D. Cosley, S. K. Lam, S. M. McNee, J. A. Konstan, and J. Riedl. Getting to know you: learning new user preferences in recommender systems. In *IUI*, pages 127–134, 2002.

[18] A. I. Schein, A. Popescul, L. H. Ungar, and D. M. Pennock. Methods and metrics for cold-start recommendations. In *SIGIR*, pages 253–260, 2002.

[19] D. H. Stern, R. Herbrich, and T. Graepel. Matchbox: large scale online bayesian recommendations. In *WWW*, 2009.

[20] X. Su and T. M. Khoshgoftaar. A Survey of Collaborative Filtering Techniques. *Advances in AI*, pages 1–20, 2009.