Where Should I Go? City Recommendation Based on User Communities

Ruhan Bidart, Adriano C. M. Pereira, Jussara M. Almeida, Anisio Lacerda
Department of Computer Science,
Universidade Federal de Minas Gerais (UFMG) - Brazil
{*ruhanbidart, adrianoc, jussara, anisio*}*@dcc.ufmg.br*

***Abstract*—Recommender systems play a key role in the decision making process of users in Web systems. In tourism, it is widely used to recommend hotels, tourist attractions, accommodations, etc. In this paper, we present a personalized neighborhood-based method to recommend cities. This is a fundamental problem whose solution support other tourism recommendations. Our recommendation approach takes into account information of two different layers, namely, an upper layer composed by cities and a lower layer composed by attractions of each city. It consists of first building a social network among users, where the edges are weighted by the similarity of interests between pairs of users, and then using this network as a component of a collaborative filtering strategy to recommend cities. We evaluate our method using a large dataset collected from TripAdvisor. Our experimental results show that our approach, despite being simple, outperforms the precision achieved by a state-of-the-art baseline approach for implicit feedback (WRMF), which exploits only the overall popularity of cities. We also show that the use of a secondary layer (attraction) contributes to improve the effectiveness of our approach.**

***Keywords*-Social Network; Collaborative Filtering; e-Tourism; Recommendation Systems**

I. Introduction

The explosive growth of content available on the Web has made the process of information search and selection an increasingly complex task. Users are often overwhelmed by the abundance of choice with which they are presented. Recommender Systems (RSs) are information filtering mechanisms devoted to overcome problems that are inherent to information overload [2].

Originally, RSs have been successfully applied on e-commerce web sites to suggest information on items and products that are related to the preferences of the user (e.g. news, web pages, movies, books, etc.). More recently, they have been also applied in the field of electronic tourism (e-tourism), providing services to assist users with travel plans [10]. A travel plan consists of a number of stages, such as: (i) choosing destinations, (ii) selecting tourist attractions, (iii) choosing accommodations, (iv) deciding routes, among others.

In this paper, we are interested in the first stage, which consists in recommending a set of cities to the user. Many travel recommendation systems focus on the second stage – suggesting a set of tourist attractions – assuming that the destination is given [10], [22], [23]. Thus, our present effort complements prior work by focusing on a more fundamental problem, which in turn supports user decisions when building a travel plan.

The key challenge in developing a system for personalized city recommendation is the integration of heterogeneous travel information. The recommendation process should take into account not only a set of different cities which are candidates for recommendation but also the attractions that would lead users to visit each city. Since cities and attractions are different types of evidence used to describe user preferences, it is difficult to automatically combine both of them into a recommendation function.

In this paper, we present a system for personalized city recommendation that takes into account the user preferences for cities and the attractions (e.g., restaurants, movie theaters, etc.) available in each city. The proposed solution is based on *Collaborative Filtering* that relies only on past user behavior (e.g., the cities each user has visited and *liked*) and does not assume explicit profiles [12]. More specifically, we present a neighborhood-based method (NBM) that is centered on computing the relationships among users.

Traditional NBMs compute the similarities between the users/items using the co-preference information, and new items are suggested based on these similarity values [7]. Our proposed method computes the similarity between pairs of users considering the cities previously visited by those users. However, unlike traditional NBMs, it also considers how similar users rated different attractions in each candidate city to select which cities should be recommended to a target user.

Specifically, we start by building a network, in which the nodes are users and the weighted edges represent the similarities among users, considering the visited cities. We then use this network to extract communities of users that have similar interests and preferences. Finally, we recommend cities, considering information about attractions of each city, to a given user using information from the community to which the user belongs.

We evaluate our recommendation method, called ReCWEE – ***Re****commendation using* ***C****ommunities and*

978-1-4799-6953-1/14 $31.00 © 2014 IEEE
DOI 10.1109/LAWeb.2014.15

***W**ithout **E**xplicit **E**valuations* – using a large dataset collected from TripAdvisor, currently one of the most popular travel websites, with nearly 260 million unique monthly visitors[1]. In order to assess the benefits of using information regarding attractions for recommending cities, we consider two variations of ReCWEE: one exploits mainly information regarding cities, whereas the other fully exploits both layers by aggregating information about cities and attractions in each city. We refer to the latter as ReCWEE+. We compare both methods against a state-of-the-art baseline, the Weighted Regularized Matrix Factorization (WRMF) [9]. Our experimental results show that ReCWEE improves the precision of the recommendations in up to 6.8%, on average, which can be attributed mainly to the use of communities. Further improvements (5.2%, on average) can also be obtained when our approach fully exploits both cities and attractions for users with not so sparse profiles.

The rest of this paper is organized as follows. Related work is discussed on Section II. In Section III, we define the problem of personalized city recommendation. We describe the proposed solution in Section IV. Our experimental methodology and main results are presented in Sections V and VI. Section VII offers conclusions and directions for future work.

II. Related Work

The idea behind recommender systems is to automatically recommend items for an user, aiming to predict the user's interest level about the items [21]. These systems help users to deal with information overload, providing personalized recommendations of products and services [2].

Recommender systems are typically classified in three categories: *content-based*, which recommends items that are similar to items that the user preferred in the past [15]; *collaborative*, which recommends items that users with similar profiles have preferred in the past [7]; and *hybrid* approaches, which combine those two to make recommendations [3].

Collaborative Filtering (CF) is one of the most successful recommendation strategies to date [17], and is used in many domains, such as social streams [5] and movies [9]. This approach represents the state-of-the-art among recommendation techniques, and it is used as the basis of our proposed method. There are two primary types of recommendation methods that are based on CF: *neighborhood methods* and *latent factor methods*. We briefly discuss prior work in these two categories next.

A. Neighborhood Methods

Neighborhood methods are centered in computing the similarities among users, and the user/item preferences are directly used to suggest new items [12]. The key problem in these methods is identifying groups of users with similar tastes. There are several techniques to identify communities in graphs [6], the simplest and most generic of them is the k-NN ($k-$Nearest Neighbor), which is used in several works [4], [8].

Formally, the utility $u(c, s)$ of an item s to an user c is based on the utilities $u(c_j, s)$ assimilated to the item s, by users c_j that are "similar" to user c. Although this task is well defined, there are several ways to tackle this problem. In [7], a framework to implement a solution to this problem is defined, which is used as basis for our proposed method. The implementation of this framework is detailed in Section V.

The aforementioned technique can be divided into three main components: (a) similarity computing, (b) neighborhood selection, and (c) top-N recommendation. The literature is rich in various efforts to develop and test algorithms that implement each of these components. For example, in [6], the authors describe and discuss various techniques to compute similarity between different users. The neighborhood selection step was addressed primarily by [7], [11], and more recently by [21], where the authors applied it in a collaborative system using a real social network (Last.FM). Solutions to the top-N recommendation step are specially developed in [8], where several techniques are empirically analyzed. Moreover, in [4], the authors compare evaluation metrics to the top-N recommendation task, which consists in recommending only the best N items that would be suited to the user.

B. Latent Factor Methods

Latent factor methods, such as Singular Value Decomposition (SVD), consist of an alternative approach that maps both items and users into the same latent factor space, thus making them directly comparable [12]. Such methods have become popular by combining good scalability with accurate predictive power.

A representative example of latent factor model is the Weighted Regularized Matrix Factorization (WRMF) [9]. This algorithm is considered the state-of-the-art matrix factorization model when considering implicit feedback. Since in our scenario users do not explicitly state their preference, we use WRMF as a strong baseline for our neighborhood based approach. In WRMF users and items are mapped in a join latent factor space of dimensionality f, such that user-item interactions are modeled as inner products in that space. Each item i is associated with a vector $q_i \in \mathbb{R}^f$ and each user u is associated with a vector $p_u \in \mathbb{R}^f$. The resulting scalar product $q_i^T p_u$, captures the interaction between u and i, leading to the estimation $\widetilde{r}_{ui} = q_i^T p_u$. The challenge here is to compute the mapping of each item and user factor vectors ($q_i, p_u \in \mathbb{R}^f$). With these values calculated we can easily estimate the rating that an user will give to any item.

[1] http://www.tripadvisor.com/PressCenter-c6-About_Us.html

The estimations of q_i and p_u are calculated minimizing the following equation [13]:

$$\min_{q*,p*} \sum_{(u,i)\in K} (r_{ui} - q_i^T p_u)^2 + \lambda(||q_i||^2 + ||p_u||^2) \quad (1)$$

Latent factor models have been successfully applied in different scenarios. For instance, in [18] the authors apply it to the recommendation of products in an e-commerce website, whereas in [14] the authors use existing location data in photos to infer user travel routes and, then, recommend routes to other users. Other prior studies that exploited WRMF are [16], [20], where the authors aim at recommending attractions[2]. Here we use WRMF as a baseline for our neighborhood-based approach, but we focus on recommending cities, while the focus of other works is in the recommendation of attractions. As far as we know, this is the first work that focus on city recommendation by exploring both user preferences for cities and also attractions associated to these preferred cities.

III. Problem Statement

We here propose a collaborative filtering approach to recommend cities to a given user. To that end, we address three main problems.

The first problem is the inference of the weights assigned to links connecting pairs of users. This problem is described as: let U = $\{u_1, u_2,...,u_n\}$ be the set of all users, and $\mathbf{T}$ $\in \mathbb{R}^{nxn}$ a matrix representing similarities between them. Our goal is to fill matrix $\mathbf{T}$, assigning weights to specific pairs of users, which ends up generating a virtual social network between users.

The second problem is to split users into groups (communities). It can be defined as: given an user u (target of recommendation) and a similarity matrix $\mathbf{T}$, the task is to find a group G_u of users that would better describe u (i.e., that are more similar to u). Note that this task is analogous to the task of community detection.

The third problem is a learning to rank task, whose goal is to rank a list of candidate cities for each user. The problem can be defined as follows. Given a target user $u \in U$ that has a group of related users G_u = $\{u_1, u_2,..., u_m\}$, the set C_g = $\{c_1, c_2,..., c_k\}$ of all cities visited by users in G_u, and the set C_u = $\{c_1, c_2,..., c_l\}$ of cities visited by u, our task consists of ranking the set of cities in C_g - C_u based on the usefulness, as a recommendation, of each city to u. Note that we only recommend cities that have not been visited by the target user.

IV. Proposed Solution

The solution we propose to address the city recommendation problem is called ReCWEE – ***Re****commendation using* ***C****ommunities and* ***W****ithout* ***E****xplicit* ***E****valuations*. The hypothesis behind it is that users who visited a large number of cities in common with target user u have a higher probability of visiting other cities that u would like to visit. In other words, the best candidate cities for recommending to u are among those visited by users with similar interests.

The design of ReCWEE assumes that there is no explicit evaluations of cities. This is the case of, for instance, TripAdvisor, which is currently the world's largest travel site according to comScore[3]. It assumes, however, that a set of cities previously visited by each user u, C_u, is given, as is the case of TripAdvisor. The algorithm uses a top-N recommendation approach, taking into account only the group of cities $C_g - C_u$ (see section III).

ReCWEE is composed of three main modules, which work independently and in series, as illustrated in Figure 1. The following sections describe each step of our solution in detail. We finish this section putting them together to build our recommendation strategy.

Figure 1. Overview of our approach. Step 1: a graph of users is created. Step 2: communities in the graph are detected. Step 3: a ranking of cities is generated and the top-k cities are recommended to the user.

A. Graph Generation

The first step in the process of graph generation is related to the question: How can we create a social network linking users? Or, in other words, how can we infer similarities between pairs of users? Since we assume that no explicit user rating about cities is available, we use the set C_u of cities previously visited by user u as evidence of u's interests. Specifically, we estimate the similarity between two users u_1 and u_2 by the similarity between their sets of visited cities, C_{u1} and C_{u2}, computed using the Jaccard coefficient [19] as follows:

$$J(C_{u1}, C_{u2}) = \frac{|C_{u1} \cap C_{u2}|}{|C_{u1} \cup C_{u2}|} \quad (2)$$

Note that we need to calculate the Jaccard coefficient between each pair of users, thus completely filling the matrix $\mathbf{T}$. We did experiment with other similarity metrics (e.g., *cossine similarity*) but the results produced with the Jaccard coefficient are at least as good as those obtained with the other metrics. We thus opted for using Jaccard because, besides the good performance, it is simple and easy to interpret.

[2]In the tourism domain, attractions are locations that can be visited in a city (e.g., museums, movie theaters, etc.).

[3]http://www.comscore.com.

Next, we have to define a threshold τ of minimum similarity. All links with weights below τ will be pruned (removed from **T**), and only the remaining links will be passed to the next step (community detection). Moreover, users that become disconnected from the rest (i.e., isolated nodes in the graph) after the link pruning will also be removed from the graph, since they will not belong to any community.

Rather than arbitrarily selecting a threshold, we opted for first analyzing the distribution of similarity values. To that end, we plotted the number of users that are removed from the graph for various values of threshold. Figure 2 shows this distribution for the dataset we use in this work, which will be presented in the next section.

Note that the distribution has a clear knee: when the threshold τ is set to 0,2, only 3,78% of the users are removed. Moreover, around 95% of the links are removed. We do want a threshold that leads to a minimum number of users removed but a large number of links removed, as those links represent weak connections between users. Thus, the choice of 0,2 seems good. We did experiment with larger values but found no improvements in recommendation precision, experimentally confirming the hypothesis raised in [7] that higher thresholds do not improve recommendation effectiveness.

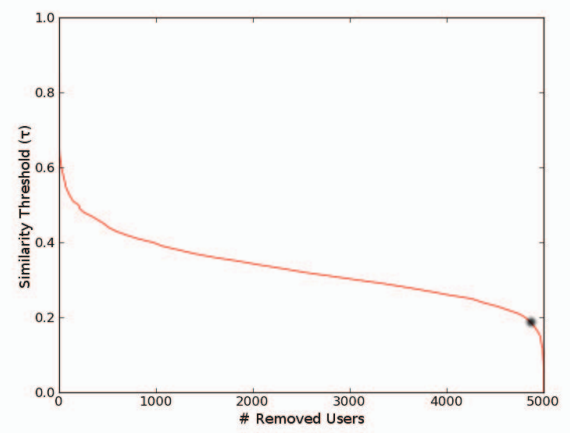

Figure 2. Number of users removed from the graph for each value of threshold. Note that the knee of the curve is around 0,2 (marked with a black circle).

B. Community Detection

In this step we use the k-Nearest Neighbor (k-NN) algorithm to infer the community of a given user u. It simply selects the k users that have the highest similarities with u as its community. Although many other community detection algorithms could be employed in this step [6], [7], we chose k-NN as it is simple and scalable to the volume of data used in this work.

The value of k was set to 20, as suggested by [7], which means that the 20 most similar users to the target user will be selected as its community. Note that user u_2 may belong to the community of u_1, while u_1 may *not* belong to the community of u_2.

The communities generated by k-NN play a key role in the next step of the method (ranking). The set of candidate cities to recommend to a target user u will be extracted from u's community. Thus, by reducing the set of neighbors to the top-20 most similar ones, we are indirectly reducing the search space for cities to be recommended to u.

C. Ranking of Candidate Cities

In this final step, the cities that are candidates to be recommended to target user u are ranked according to the *usefulness* of them to u. The candidate cities are extracted from u's community: they correspond to all cities that were visited by users in u's community (G_u) but not by u, that is, cities in set $C_g - C_u$ (see section III). The usefulness of each candidate to u is estimated by a *score* function. Given this score function, the candidates are simply ranked in decreasing order of score.

One key contribution of this work is propose a score function that uses information of two layers - cities and city attractions. To assess to which extent the use of the lower layer - city attraction - improves recommendation effectiveness, we also consider an alternative score function that exploits only information about the cities.

The simplest function assigns a score to a city c candidate to recommendation to target user u according to the number of users in u's community that had visited c in the past and the average of rankings given to all attractions of this city. That is, this score simply takes the popularity of each city in u's community (G_u), weighting each city also by the overall opinion about its attractions. This score function is defined as:

$$Score_1(c,u) = Popularity(G_u, c) \times MeanAttractionEvaluations(c) \quad (3)$$

where $Popularity(G_u, c)$ is the fraction of users in G_u that had visited c in the past.

We also propose a score function that takes into account not only the popularity of a candidate city in G_u but also how each user in G_u evaluated each of its attractions, that is:

$$Score_2(c,u) = Popularity(G_u, c) \times \sum_{u_i \in G_u} \frac{\frac{1}{|Attr_c|} \sum_{a \in Attr_c}^{n} \frac{Eval(a,u_i)}{Bias(u_i)}}{WithRating(G_u, c)} \quad (4)$$

where $Attr_c$ is the set of attractions in city c, $Eval(a, u_i)$ is the rating that user u_i gave to attraction a in a 1-5 range, and $Bias(u_i)$ is the average of all ratings given by the user u_i. We normalize the evaluations of u_i by this factor so as to better capture any bias u_i might have towards giving higher or lower ratings, and thus more accurately

group the opinions of different users in G_u regarding each attraction. The innermost summation aggregates the opinions of neighbor u_i with respect to all attractions in city c. We then divide this number by the number of attractions in c to take an average opinion of u_i regarding c, and thus avoid favoring cities with a larger number of attractions. Finally, we take the average of these opinions across all neighbors that have rated at least one attraction in the city (outermost summation, where $WithRating(G_u, c)$ gives the number of users in G_u who rated any attraction in city c) and multiply it by the popularity of the city in the group.

We note that ratings of attractions are typically very sparse (as we observed in our dataset). In particular, there might be no ratings about attractions of the city in the community of the target user, or even in the whose dataset. Thus, we propose to carefully adapt how our $Score_2$ function is applied depending on the available data. The main issue regards the set of user G_u in Equation 4. We envision three scenarios:

1) If there are ratings for attractions of the city from the user community: in this case we use the opinions of users in this community, as in Equation 4;
2) If there are no ratings for attractions of the city in the user community, but there are ratings from other users (outside the community): in this case, we replace G_u in Equation 4 by all users in the graph that have rated some attraction in that city.
3) If there are no ratings of attractions from, then our score function reduces to only evaluating the $Popularity$ of the city in G_u.

D. Putting All Together: ReCWEE

The pseudocode of $ReCWEE$ is shown in Algorithm 1. The algorithm receives as input a target user u as well as the set of all users U.

The algorithm starts by generating a graph (line 25). The function $GraphGeneration$ (lines $1-12$) creates a complete graph between users, using Jaccard index of the cities visited by each one of them, and then prunes weak links (and isolated nodes) according to a pre-defined similarity threshold τ.

Next, the k-NN algorithm is used to generate a community for target user u (line 26). Finally, the function Ranking is called to produce a ranking of the candidate cities. The top-N cities are recommended (line 27). Note that the first step, which is more costly, is done offline, at a training phase. Only the community detection and the ranking need to be performed at recommendation time.

The function $Ranking$ (lines $14-21$) gets the target user u and her/his community as parameters, and returns a ranking of cities to be recommended to this user. It first determines the candidate cities by taking cities that users in u's community have visited and removing those that u has already visited, aiming at recommending only cities that the user has not visited yet. Each city receives a score (line 17), using Equation 3 or Equation 4. The list of candidate cities sorted by score in reverse order is then returned (line 19).

Algorithm 1 ReCWEE

```
1: function GraphGeneration(U, τ)
2:     Graph ← ∅
3:     for u1 in U do
4:         for u2 in U do
5:             if u1 ≠ u2 then
6:                 Graph[u1][u2] ← Jaccard(Cities(u1), Cities(u2))
7:             end if
8:         end for
9:     end for
10:    Prune Graph according to threshold τ
11:    return Graph
12: end function
13:
14: function Ranking(u, Gu)
15:    CandidateCities ← Cities(Gu) − Cities(u)
16:    for c in CandidateCities do
17:        c ← score(c, u) ▷ Equation 3 (ReCWEE) or Equation 4 (ReCWEE+)
18:    end for
19:    rank ← SortReverseOrder(CandidateCities, score)
20:    return rank
21: end function
22:
23: function ReCWEE(u, U)
24:    τ ← 0, 2
25:    Graph ← GraphGeneration(U, τ)
26:    Gu ← GenerateCommunitiesKNN(u, Graph)
27:    return top-N in Ranking(u, Gu)
28: end function
```

In order to distinguish between the two proposed score functions (Equations 3 or 4), we refer to our recommendation approach exploiting both cities and attractions (i.e., using Equation 4) as ReCWEE+. It is important to note that, though ReCWEE+ is explained based on the *cities* x *attractions* scenario, it can be easily generalized to other scenarios that have hierarchical structure. For example a system where user gives ratings to answers and you want to recommend categories of answers (*categories* x *answers* scenario).

V. Experimental Methodology

In this section, we first present our dataset and the data acquisition process (Section V-A). We then present the methods used as baselines in our evaluation (Section V-B), and briefly describe our evaluation setup (Section V-C).

A. Data Acquisition

The dataset used in this work was collected from TripAdvisor [1], and covers a period of over 5 months, from October 23^{rd} 2013 to March 23^{rd} 2014. We focus on recommending cities to Brazilians. Thus, as a starter to our crawling process, we select two groups of seeds: one containing the 30 most popular touristic cities in Brazil, and the other with the 15 most popular touristic cities in the world. These rankings of most popular cities are taken from TripAdvisor's Web site.

Starting with the set of seeds, our crawler proceeds as follows: it first collects the city page and then gathers

each attraction found in it. For each attraction, it collects all reviews posted by users regarding that attraction. After that, it downloads the profile of the review's author, which includes the list of cities previously visited by this user. Finally, it repeats the process for each new city discovered, as illustrated in Figure 3.

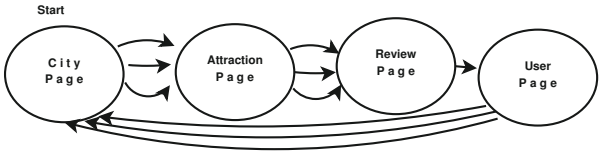

Figure 3. Overview of our TripAdvisor crawler. It starts with a set of cities (seeds), visits all attractions that are listed on it. After that, it collects all reviews of each attraction. For each review it collects the user that wrote the review. Finally, it get all cities visited by the user and follows it, restarting the process. Transition with one arrow means that the *from* page has only one entity that leads to the *to* page, while in transitions with more than one arrow, the page *from* leads to several entities *to*.

Table I summarizes our dataset, presenting the numbers of the main entities – pages, cities, attractions, reviews and users – available. We here focus on three of those entities, namely *Cities*, *Attractions* and *Users*. We leave the design of recommendation strategies that also exploit user reviews as a future work. It is important to emphasize that all users were anonymized for privacy reasons. For each user, our dataset contains an identifier, a location, a list of cities she/he has visited (city *seenlist*), a list of attractions she/he has rated (attraction *seenlist*). Each attraction is associated with a name, an address, a city, an average user rating, and a number of reviews.

Table I
OVERVIEW OF OUR DATASET COLLECTED FROM TRIPADVISOR

Entity	Number
Pages	1,597,609
Cities	85,505
Attractions	162,168
Reviews	599,629
Users	266,392

B. Baseline Methods

We compare our proposed methods against the following two baselines:

1) **Popularity**: this is a very basic and intuitive strategy that always recommends the most popular cities, regardless of the target user. In other words, it is a non-personalized recommendation strategy. The popularity of a city is estimated by the number of users who have previously visited it, i.e., the number of users who have that particular city in their city seenlists. This strategy, though naive, has shown to be surprisingly powerful in various applications [9];
2) **Weighted Regularized Matrix Factorization (WRMF)**: this algorithm is considered to be the state-of-the-art for implicit feedback recommendation, therefore it is a strong baseline and difficult to outperform. In our case, we modeled WRMF with users and cities (items). WRMF receives the relations between users and cities (implicit), discovers latent factors about them and, after that, uses these latent factors to recommend cities to the users (for more details about WRMF see Section II-B).

C. Evaluation Setup

In our study, we use a fully automatic evaluation methodology, as most previous work [4], [7]–[9], [14]. Specifically, we adopt a five-fold cross-validation procedure. That is, the dataset is randomly split into 5 pieces (folds), each one containing 20% of the *cities* and associated information (attractions and users). Four folds are used for training the recommendation methods, i.e., for computing the similarities between users as well as the popularity of cities, and one fold is used as test set to evaluate the methods. The folds are switched and the process is repeated 5 times, each one with different training and test sets. We note that, in the particular case of ReCWEE and ReCWEE+, only data in the training set is used in the graph generation phase. Thus, only cities in the training set can be considered candidates for recommendation.

The test sets are used to evaluate the recommendations. A city is considered a relevant recommendation to a target user u if it appears in the city seenlist of u (that is, we do have evidence that u visited the city) and is in the *test* set[4], that is, from the perspective of the recommendation strategy, it is unknown.

We use **precision** and **recall** in the top-k recommended cities as main evaluation metrics. Let C_u be the sorted list of recommended cities produced by the method being evaluated to a target user u, and C_u^k the top k cities in this list. Let also Rel_u be the set of relevant cities for user u (extracted from the test set). The precision in the top-k recommendations, p_k, is defined as:

$$p@k(C_u^k, Rel_u) = \frac{|C_u^k \cap Rel_u|}{min(k, |C_u^k|)} \quad (5)$$

The recall in the top-k recommendations is defined as:

$$recall@k(C_u^k, Rel_u) = \frac{|C_u^k \cap Rel_u|}{min(k, |Rel_u|)} \quad (6)$$

All tests were executed five times with a different selection of users in each of them. It is reported the confidence interval (95%) of these results. Following, we present the results.

[4]Otherwise, if it is in u's seenlist, it must be in the training set and thus is not considered as a candidate for recommendation.

VI. Results

This section presents results of the experiments, applying our two proposed algorithms (ReCWEE and ReCWEE+) in a real dataset collected from TripAdvisor, and comparing these results with two baselines. At first, we are not worried about comparing performance of the algorithms, because all the process must be executed offline and only once a day. An important point to be emphasized about our experiments is related to the number of cities on users' seenlists. As expected, the distribution of the seenlists length by number of users follows a long tail distribution, as shown in Figure 4, indicating that most users have seenlists with only a few cities.

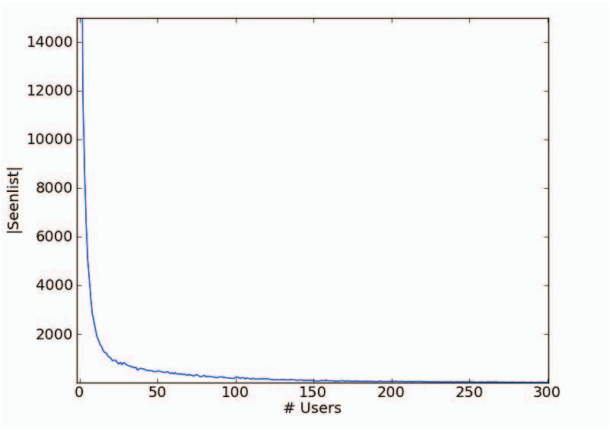

Figure 4. Distribution of seenlist's length by users.

Thus, two types of results will be presented, one for users with small seenlists and another for users with big seenlists. Section VI-A presents the experiments for a larger group of users (small seenlists) and section VI-B corresponds to an analysis of a group of users that have big seenlists.

A. Results for groups with small seenlists

As the focus of this work is not the *cold-start* problem, it was chosen to filter users that have seenlists with less than 10 cities, which resulted in 63,822 users. Nevertheless, these users are from several nationalities and it is desired to work only with Brazilians, because the objective of this work is to understand their behavior and recommend cities to them. Thus, the data was filtered to select only Brazilians, but because of unstructured data and by the lack of location data for several users, this filter resulted in 26,549 users.

It was generated rankings for the four algorithms (Popularity, ReCWEE, ReCWEE+ and WRMF) in five groups of 5,000 users, chosen randomly from this 26,549, and these groups can share users between them. For each one of these five groups, we executed tests with five-fold cross-validation. The results to these experiments are reported in Figure 5, with confidence interval of 95%. In the case of WRMF we have tried several different parameters configuration, the best of them was $\lambda = 150$ and $number\ of\ factors = 100$ (as suggested in [9]). This configuration set was used in all experiments that are shown in this section.

It is important to note that in Figure 5 the algorithms ReCWEE and ReCWEE+ are significantly better than the other two in terms of precision. We also can see that the difference between our two proposed approaches was small. In the next section we will show that it is because of the lack of data to improve recommendations. Once the sole difference between ReCWEE+ and ReCWEE is the use of attractions, when these data is very much sparse or there is no data, ReCWEE+ can not outperforms ReCWEE. Note that in neither case ReCWEE overcomes ReCWEE+, as expected.

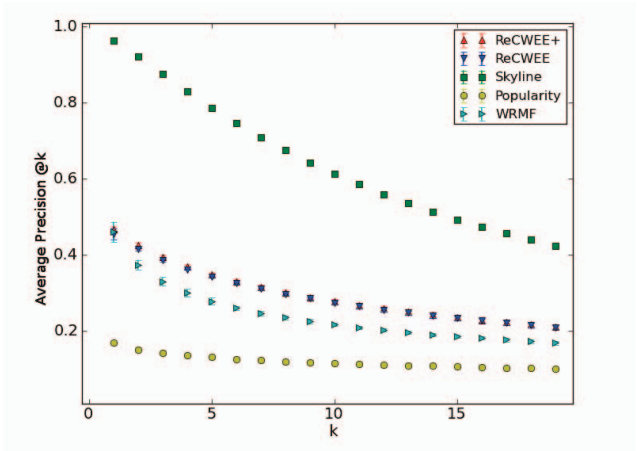

Figure 5. Precision of the four algorithms and *Skyline* (maximum possible precision for ReCWEE and ReCWEE+). All results are reported with confidence interval of 95%. Note that the algorithm ReCWEE and ReCWEE+ are significantly better than the other two, an that the difference between our proposals, with this amount of available data, is small. It occurs due to missing data about attractions and even because of small seenlists.

In Figure 5 we also report the *Skyline*. These green squares are a measure of the maximum that our approaches could reach. It is because we use a neighborhood method and it filters data by similar users before recommending. Green squares show the maximum that could be achieved using groups selected by our neighborhood selection approach. They also show that our two first steps (see section IV), which filters candidate cities, are well done, once it is observed a great distance between *Skyline* and all methods.

Recall results are presented in Figure 6. It can be seen that ReCWEE and ReCWEE+ algorithms have better recall than the other two baseline algorithms. Green squares present the maximum recall that could be reached using our neighborhood model. Note that it shows again that our method of filtering candidate cities is good because all methods are far

away from the upper limit (*Skyline*).

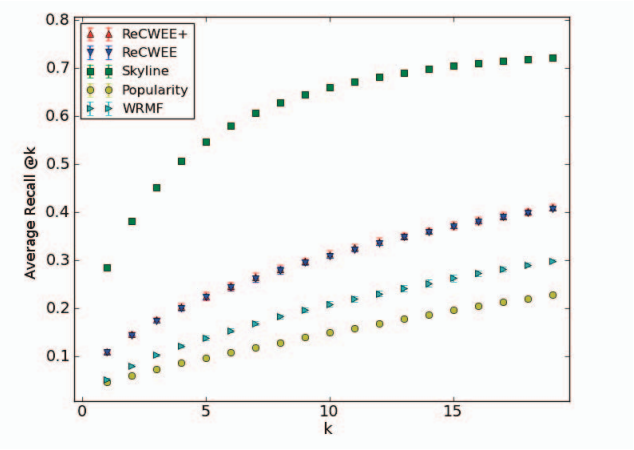

Figure 6. Recall results to the four algorithms and the *Skyline* limit for our proposed methods. All results are presented with confidence interval of 95%. Note that ReCWEE and ReCWEE+ are also significantly better than the others in terms of recall but not significantly different to each other.

B. Results for groups with big seenlists

The same tests were executed for $|seenlist| \geq 100$, aiming to check the hypothesis that ReCWEE+ should achieve better precision for users with more cities in their seenlists. The results of these experiments are presented in Figure 7.

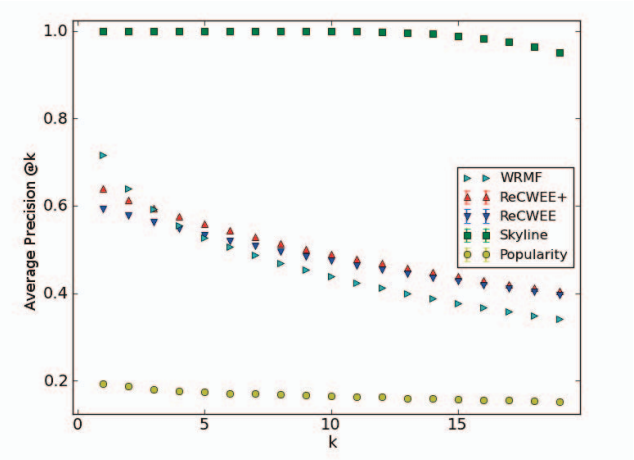

Figure 7. Precision for groups of users with big seenlists.

The most important aspect to note in Figure 7 is that with more data ReCWEE+ outperforms ReCWEE in 4.2% when considering Precision@5 (on average), confirming our hypothesis that the use of a second layer information (attractions) can improve our recommendation method.

Another important aspect is that for users with more data in their profiles WRMF outperforms ReCWEE and ReCWEE+ on the first two positions of the ranking. Although it is an advantage for WRMF, it is important to realize that it overcomes our methods only in the two first ranking positions. In all next ranking positions our proposed solution outperforms the baseline, which is very important in our case because recommending cities makes more sense when you recommend at least five cities, providing more choices to the user. Despite seeing this behavior of WRMF, in recall (see Figure 8) ReCWEE+ is always better than WRMF, in all ranking positions.

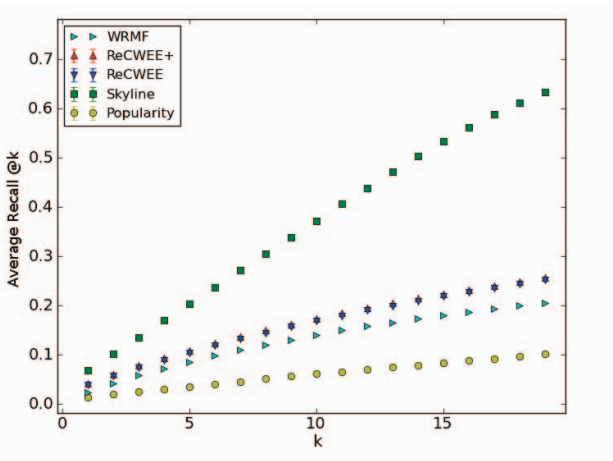

Figure 8. Recall for groups of users with big seenlists. Note that recall for ReCWEE and ReCWEE+ is better than WRMF.

The result analysis also allows to understand that it is possible to create groups containing only users with more data in their seenlist, in order to develop a hierarchical recommendation task, where groups of users with similar number of cities in their seenlists would be segmented and used together to get a better precision using only their relations (reduction in sparsity by removing users with small seenlists).

VII. Conclusion

Recommendation is a way of helping users in the increasingly difficult task of making decisions on-line, whatever the nature of the decision. In this work, the task of recommending cities was faced for the first time, as far as we know. This task is shown as being important for the simple fact that there are systems, such as TripAdvisor, where people share their opinions about the most different destinations with the goal of helping each other to make decisions about traveling.

First, we formalize the problems addressed in this paper. Then, we detail a methodology to solve each one of these problems. The proposed methodology, although it has been applied in an actual case study, is not specific and can be

used in different scenarios, establishing itself an important contribution of this work.

Our results showed that infer a social network on TripAdvisor, along with detecting communities on it, is a good technique to improve precision in recommendation of cities that users have not visited yet.

The results also show that the use of a secondary data layer (attractions) brings benefits to our method, increasing substantially precision in comparison with the method that does not use a lower layer. ReCWEE+ also outperforms the state-of-the-art method (WRMF) for users with a small data volume and, for those with big volume of data, ReCWEE+ overcomes WRMF after the position @3 in the ranking.

Finally, the findings show that techniques of segmentation and hierarchy of users could be used to further improvements in precision, since the split of users with a higher information showed a higher precision.

As future work it is intended to improve the currently ranking technique, taking into account the diversity of recommendations. It is also intended to use data from reviews of each attraction in order to find out more descriptive characteristics of cities, making it possible to group cities by its similarities, which would enable to recommend cities by groups not only individually. Additionally, it is intended to link users by a real social network, using data from $Facebook$ that are available for a representative set of users of our TripAdvisor's dataset.

ACKNOWLEDGMENT

This research was supported by the Brazilian National Institute of Science and Technology for the Web (CNPq grant numbers 573871/2008-6 and 477709/2012-5), CAPES, CNPq, Finep, and Fapemig.

REFERENCES

[1] "Tripadvisor," http://tripadvisor.com.br, 2014.

[2] G. Adomavicius and A. Tuzhilin.

[3] M. Balabanović and Y. Shoham, "Fab: Content-based, collaborative recommendation," *Commun. ACM*, vol. 40, no. 3, pp. 66–72, Mar. 1997.

[4] P. Cremonesi, Y. Koren, and R. Turrin, "Performance of recommender algorithms on top-n recommendation tasks," in *Proceedings of the 4th ACM Conference on Recommender Systems*, 2010, pp. 39–46.

[5] E. Diaz-Aviles, L. Drumond, L. Schmidt-Thieme, and W. Nejdl, "Real-time top-n recommendation in social streams," in *Proceedings of the Sixth ACM Conference on Recommender Systems*, ser. RecSys '12, 2012, pp. 59–66.

[6] S. Fortunato, "Community detection in graphs," *Physics Reports*, vol. 486, no. 3-5, pp. 75 – 174, 2010.

[7] J. Herlocker, J. A. Konstan, and J. Riedl, "An empirical analysis of design choices in neighborhood-based collaborative filtering algorithms," *Inf. Retr.*, vol. 5, no. 4, pp. 287–310, Oct. 2002.

[8] J. L. Herlocker, J. A. Konstan, L. G. Terveen, John, and T. Riedl, "Evaluating collaborative filtering recommender systems," *ACM Transactions on Information Systems*, vol. 22, pp. 5–53, 2004.

[9] Y. Hu, Y. Koren, and C. Volinsky, "Collaborative filtering for implicit feedback datasets," in *Proceedings of the Eighth IEEE International Conference on Data Mining*, ser. ICDM '08, 2008, pp. 263–272.

[10] Y. Huang and L. Bian, "A bayesian network and analytic hierarchy process based personalized recommendations for tourist attractions over the internet," *Expert Systems with Applications*, vol. 36, no. 1, pp. 933–943, 2009.

[11] M. Jamali and M. Ester, "Using a trust network to improve top-n recommendation," in *Proceedings of the 3rd ACM Conference on Recommender Systems*, 2009, pp. 181–188.

[12] Y. Koren, "Factorization meets the neighborhood: a multifaceted collaborative filtering model," in *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2008, pp. 426–434.

[13] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, 2009.

[14] T. Kurashima, T. Iwata, G. Irie, and K. Fujimura, "Travel route recommendation using geotags in photo sharing sites," in *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*, ser. CIKM '10, 2010, pp. 579–588.

[15] G. Linden, B. Smith, and J. York, "Amazon.com recommendations: Item-to-item collaborative filtering," *IEEE Internet Computing*, vol. 7, no. 1, pp. 76–80, Jan. 2003.

[16] A. Noulas, S. Scellato, N. Lathia, and C. Mascolo, "A random walk around the city: New venue recommendation in location-based social networks." in *SocialCom/PASSAT*. IEEE, 2012, pp. 144–153.

[17] F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, Eds., *Recommender Systems Handbook*. Springer, 2011.

[18] J. B. Schafer, J. A. Konstan, and J. Riedl, "E-commerce recommendation applications," *Data Min. Knowl. Discov.*, vol. 5, no. 1-2, pp. 115–153, Jan. 2001.

[19] P.-N. Tan, M. Steinbach, and V. Kumar, *Introduction to Data Mining*. Addison Wesley, 2006.

[20] H. Wang, M. Terrovitis, and N. Mamoulis, "Location recommendation in location-based social networks using user check-in data," in *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, 2013, pp. 374–383.

[21] X. Yang, H. Steck, Y. Guo, and Y. Liu, "On top-k recommendation using social networks," in *Proceedings of the Sixth ACM Conference on Recommender Systems*, ser. RecSys '12, 2012, pp. 67–74.

[22] M. Ye, P. Yin, W.-C. Lee, and D.-L. Lee, "Exploiting geographical influence for collaborative point-of-interest recommendation," in *Proceedings of the 34th international ACM SIGIR Conference on Research and development in Information Retrieval*, 2011, pp. 325–334.

[23] Q. Yuan, G. Cong, Z. Ma, A. Sun, and N. M. Thalmann, "Time-aware point-of-interest recommendation," in *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2013, pp. 363–372.