Tsakalakis and Koutsakis
Hum. Cent. Comput. Inf. Sci. (2018) 8:36
https://doi.org/10.1186/s13673-018-0160-7

Human-centric Computing and Information Sciences

RESEARCH Open Access

Improved user similarity computation for finding friends in your location

Georgios Tsakalakis[1] and Polychronis Koutsakis[2*]

*Correspondence:
p.koutsakis@murdoch.edu.au
[2] School of Engineering and Information Technology, Murdoch University, Science and Computing Building 245, SC1.012, 90 South Street, Murdoch, WA 6150, Australia
Full list of author information is available at the end of the article

Abstract

Recommender systems are most often used to predict possible ratings that a user would assign to items, in order to find and propose items of possible interest to each user. In our work, we are interested in a system that will analyze user preferences in order to find and connect people with common interests that happen to be in the same geographical area, i.e., a "friend" recommendation system. We present and propose an algorithm, Egosimilar+, which is shown to achieve superior performance against a number of well-known similarity computation methods from the literature. The algorithm adapts ideas and techniques from the recommender systems literature and the skyline queries literature and combines them with our own ideas on the importance and utilization of item popularity.

Introduction

The diversity of social networks makes the problem of correctly estimating user preferences essential for personalized applications [1]. Most recommender systems suggest items of possible interest to their users by employing collaborative filtering to predict the attractiveness of an item for a specific user, based on the user's previous rating and the ratings of "similar" users. In this work, rather than focusing on possible items of interest for a user, we are interested in designing an algorithm that will utilize user ratings in order to recommend one user to another as a possible friend.

This paper continues our recent work [2], where we presented the architectural design, the functional requirements and the user interface of eMatch [3], an Android application which was inspired by the idea of finding people with common interests in the same geographical area. Close friendship is a measure of trust between individuals [4] and friends have the tendency to share common interests and activities, as has been shown in multiple studies in the literature starting with important work on personality similarities and friendship dating in the 70 s [5, 6] and continuing until today [7]. In terms of social networks, people selectively establish social links with those who are similar to them, and their attitudes, beliefs and behavioral propensities are affected by their social ties [8, 9]. In eMatch, in order to compare people's interests, users rate as many as nine interest categories: {Movies, Music, Books, Games, Sports, Science, Shopping, Food, Travel}, while they can add and rate items to each one of them. For example a user could rate the category "Sports" with "7" on a scale of 1 to 10, and add to this category the item "football"

with rating "9". Based on this type of rating, the application's algorithm computes users' matching in order to suggest potential friends.

The information location is used in eMatch only for practical reasons, i.e., in order to locate potential friends in the same area and not for tracking on the map and revealing the user's location as other applications do. The goal of eMatch is to facilitate potential friends to meet and introduce themselves to each other only if they so wish. The user's location is considered private and sensitive information and is treated that way. The only information that is public is the matching percentage for all pairs of "Visible" users inside the geographical area. In this way the individual's privacy is preserved. More information can be found in [2].

The work in that paper introduced EgoSimilar, an algorithm which computes the similarity between users and is implemented in eMatch. Based on a dataset of 57 users, EgoSimilar was found to outperform two of the most well-known similarity measures, the Pearson Correlation and the Cosine Similarity, in regard to the most significant metrics used our study. Unlike other approaches in the literature, EgoSimilar takes into account the popularity of the items that have been rated in its computations.

The main contributions of the present work are as follows. We collected a much larger number of completed questionnaires (286 in total) from users of eMatch and evaluated EgoSimilar again, in order to study whether the conclusions of the work in [2] were confirmed. After confirming the excellence of EgoSimilar again in comparison to the Pearson Correlation and Cosine Similarity, however, we added into our new study several similarity measures, one of which was found to outperform EgoSimilar. For this reason, we substantially changed EgoSimilar by adapting ideas and techniques from the recommender systems literature and the skyline queries literature. The new algorithm, Egosimilar+, presented in this paper for the first time, is compared against several of the most well-known similarity computation methods from the literature and is shown to outperform all of them in regard to being able to identify existing friendships.

The rest of the paper is structured as follows. In "Related work" section we discuss related work in the field. "EgoSimilar" section briefly presents the original EgoSimilar algorithm. In "Evaluation of Egosimilar" section we evaluate EgoSimilar versus other similarity computation methods. "EgoSimilar+" section presents EgoSimilar+ and discusses the ideas and the motivation behind the new algorithm. "Evaluation of EgoSimilar+" section presents the results with the use of EgoSimilar+. Finally, "Conclusions and future work" section presents the conclusions of our study and the next steps in our work.

Related work

The Youhoo application [10] is the closest to eMatch among all current applications in iOS and Android that are related to finding friends in an area near the user. Its goal is to create circles of people with common interests in an area. However, Youhoo profiles are created from Facebook, therefore users who do not use Facebook are excluded from using the application, and users who wish to create a different profile in their Youhoo profile cannot do so. Additionally, the circles of people with common interests created by the application are quite generic or one-dimensional, e.g., students in the same university, people working in the same field, fans of a specific singer. On the contrary, eMatch

computes the match between users based on the whole profile that the users wish to share through the application, and of course allows users to create a profile that is independent from any other application. Other social networking applications like GeoSocials [11] and Jiveocity [12] simply present to the user other users in the same location, without any recommendation on whether they would be a good match as friends.

Other proposals from the literature for friend recommendation focus on link prediction utilizing node proximity [13], on recommending which Twitter users to follow based on user-generated content which indicates profile similarity [14], on recommending friends according to the degree to which a friend satisfies the target user's unfulfilled informational need [15], and on selecting a community of users that can meet the specific requirements of an application [16]. The work in [16] differs from our work not only because of its different goal, but also because it computes a metric based on a binary characterization of users' interest in a specific item (interested/not interested) which does not include information on the degree of user interest for the item; it also does not consider common interest categories between users as our work does, therefore related interests are considered to be completely different.

The authors in [17] use ranking functions to propose a method that represents people's preferences in a metric space, where it is possible to define a kernel-based similarity function; they then use clustering to discover significant groups with homogeneous states. The authors point out the success of the Pearson Correlation and the cosine similarity in order to make comparisons between the rating vectors of different users and they use cosine similarity in their work. As it will be shown in our results, our proposed similarity computation approach outperforms both the Pearson Correlation and the Cosine Similarity. Also, the proposed class separation technique in [17], which utilizes Support Vector Machines, becomes computationally complex and leads the authors to avoid using K-means clustering, to decrease the computational complexity of combining K-means with their technique.

EgoSimilar

In this section we present our "matching" algorithm from [2], EgoSimilar, for computing the similarity between users based on their interests and preferences. We also present, briefly, all the other widely used methods for assessing similarity that we will compare to EgoSimilar. All approaches were implemented in eMatch, in order to find potential friends based on user ratings. These algorithms run from the server side in order to keep the computational cost contained. Their running at the smartphone would be uneconomic, and also battery- and time-consuming, since it would constantly require data transfers via mobile internet and many calculations to be executed.

For the matching algorithm to run at the server, the mobile must have Internet access and at least one location provider activated. It should also store periodically (e.g., every 10 min) the geographical location of the user.

EgoSimilar takes the following rationale into account:

a. The matching is done in an "egocentric" way because each user should search friends based on his/her own criteria and interests. Thus, the matching percentage between two users that will appear on each user's screen will most likely be different. Hence,

if for example user X has one active category of interest while user Y has five, the matching percentage (X, Y) will be based on that one category, while the matching percentage (Y, X) on all five, leading to different results showing on each user's screen.

b. More popular items (popular in the sense that they are rated positively or negatively by many users) should not affect matching results as much as less popular items do, if users "agree" on them. The reason is that even if users share, e.g., a favorable opinion on a very well-known band, book, movie, etc., this does not really give a substantial hint that their tastes match in general. A similar case regarding a relatively unknown band/book/movie gives a much stronger indication of common interests. This was also pointed out in [18], where it was explained that the presence of popular objects that meet the general interest of a broad spectrum of audience may introduce weak relationships between users and adversely influence the correct ranking of candidate objects. The work in [18], however, is different from ours, as it begins with the construction of a user similarity network from historical data, in order to calculate scores for candidate objects. Our work in this paper focuses on recommending people as potential friends, not items of interest, and no historical data is relevant due to the nature of our study.
c. The rating choices of users are on a scale from 1 to 10. Consequently the maximum rating difference will be 9 and the weight of one unit in rating difference will be $1/9 \approx 0.11$. This weight is included in the computation of the similarity between users.

The steps followed by eMatch in computing the matching between users are described below. The first three steps are followed regardless of the matching computation method, which is implemented in step 4.

Let X be the user who runs the application, therefore, the matching is done according to X's tastes.

1. Check if the user's location is stored. If not, inform the user, else go to the next step.
2. Find users that are in close geographical proximity with user X.
3. Find all the active interest categories of user X.
4. The matching in EgoSimilar is computed as follows: for each user Y found in step 2, calculate the

$$\text{Matching}(X,\ Y) = \frac{1}{k_X}\sum_{c=1}^{k_X}\left[w_1[1-0.11d_1(X,Y,c)] + \frac{w_2}{n_x^c}\sum_{i=1}^{n_x^c}[1-0.11d_2(X,Y,c,i)]\right] \tag{1}$$

where k_X is the number of active categories of user X, $k_X \in [1, 9]$; w_1 is the weight attributed to the general rating of a category; w_2 is the weight of the ratings of all individual items of a category. In our case, w_1 should be smaller than w_2, as we consider the "general" matching of users (e.g., both of them loving movies), to be of smaller importance, as their specific tastes in that category may differ significantly or even completely. The exact values of w_1 and w_2 are discussed in "Evaluation of

EgoSimilar" section; n_X^c is the number of items user X has inserted in category c; $d_1(X,Y,c)$ is a function which computes the absolute difference in ratings between users X and Y for the cth activated category of user X. If user Y has deactivated the specific category, then we set $(1-0.11\cdot d_1(X,Y,c))=0$; $d_2(X,Y,c,i)$ is associated with the ith item inserted by user X in cth activated category and denotes the distance of ratings between users X and Y for the specific item.

- If user Y has not rated this item, then we set $(1-0.11\cdot d_2(X,Y,c,i))=0$,
- Otherwise $d_2(X,Y,c,i)$ is calculated, taking into account the popularity of the specific item, as follows:

a. Compute $d_2(X,Y,c,i)$.
b. Let m be the number of users that have inserted this item, and n be the number of users that have inserted items in the cth activated category of user X. Then, the popularity weight of the specific item is defined as: $W_i^c(X)=m/n$. An item is assumed to be popular if $W_i^c(X)>0.5$, which means that more than half of the users that "voted" for this category have inserted the specific item (with either negative or positive rating).
c. $d_2(X,Y,c,i)$ is adapted with respect to the popularity of the item and the rationale explained above, as follows:

If $(W_i^c(X)>0.5$ AND $d_2(X,Y,c,i)<5)$, then
$d_2(X,Y,c,i)=d_2(X,Y,c,i)+W_{change}\cdot d_2(X,Y,c,i)$.
else if $(W_i^c(X)>0.5$ AND $d_2(X,Y,c,i)\geq 5)$, then
$d_2(X,Y,c,i)=d_2(X,Y,c,i)$
else if $(W_i^c(X)\leq 0.5$ AND $d_2(X,Y,c,i)<5)$, then
$d_2(X,Y,c,i)=d_2(X,Y,c,i)-W_{change}\cdot d_2(X,Y,c,i)$.
else if $(W_i^c(X)\leq 0.5$ AND $d_2(X,Y,c,i)\geq 5)$, then
$d_2(X,Y,c,i)=d_2(X,Y,c,i)+W_{change}\cdot d_2(X,Y,c,i)$.

This states that when an item is popular and the ratings of users are close, then this item should not affect matching results as much as less popular items do. Therefore, the distance of the ratings between users X and Y must be increased in order to decrease their matching. This increase is implemented via the W_{change} weight, the value of which is discussed in "Evaluation of EgoSimilar" section.

If, however, the item is popular and the ratings of users are close, then this item should affect matching results more than the popular items do. Accordingly, the distance of the ratings between users X and Y must be decreased in order to increase their matching. This is implemented once again via the W_{change} weight.

Similarly, in the case where the item is not popular and the ratings of users are not close, we infer that this is an indication of users that do not have common interests. So, by increasing the distance of their ratings, their matching is decreased.

The complexity of the algorithm is: $O(pqr)$, where p is the number of the users, q is the number of categories (in our case, nine) and r is the maximum number of items inserted in one of the categories.

Evaluation of EgoSimilar

For our preliminary evaluation we wanted to confirm whether the results presented in [2] would stand for a much larger dataset and to investigate whether EgoSimilar would also excel in comparison with several additional similarity computation measures.

We collected data from 286 users (in comparison to the 57 users in our previous work), ages 18–40. Of the 286 participants, 272 had at least one connection (i.e., were friends) with a person from our dataset in real life. The collected information consisted of the activation/deactivation of the 9 interest categories, the Ratings for all active categories and the Ratings for the individual items in all the active categories. The items rated in each category were either new insertions by the users or as many of the default items as the users wished to rate. The mean rating given by the users was 6.6 and the standard deviation 2.5. These statistics confirmed the tendency shown in [2], of users mainly rating items that they like instead of taking the time to also add several items that they dislike. The details of the dataset are presented in Table 1.

The reason we chose to collect data mainly from groups of friends, a choice which carries a bias in the dataset and the results, was that in this way it would be feasible to evaluate whether the similarity computation methods would be able to "discover", through higher matching values, existing friendships.

To compare the results we ran the K-means clustering algorithm, each time with a different similarity computation measure (EgoSimilar and five other measures that are presented later in this section). We derived results for a number of clusters K ranging from 5 to 20 in order to evaluate how (and if) the number of clusters influences the user matching. K-means is preferred in comparison to new efficient methods like the one proposed in [19], because we do not want to have predefined classes in our system; classes (clusters) need to change based on the users who find themselves in the same area. Also, contrary to several approaches where it is useful to have weighted information incorporated into similarity scores (e.g. [20]), in our system all users should have equal weights when computing their similarity.

The following metrics and parameters were used in our study (the abbreviations are also presented in Table 2 for ease of reading):

a. Average friends' placement (AFP). This is arguably the most important metric of all, in terms of evaluating the quality of a similarity metric, as it refers to the order in which "matching users" appear on the user's screen, in decreasing percentages. A user would obviously consider first the users with whom he/she has the high-

Table 1 Dataset

Number of participants	286
Ages	18–40
Gender	177 male, 109 female
Mean rating	6.6
Standard deviation	2.5
Number of participants with at least one connection	272

Table 2 Abbreviations

Abbreviation	Meaning
AFP	Average friends' placement
N1	Number of users in the cluster
N2	Number of users in the cluster with a network
N3	Number of users in the cluster connected in reality as friends
AVC	Average valid connections
AM	Average matching
AMC	Average matching of connected users
AMnC	Average matching of not connected users

est matching, regardless of the actual matching percentage (unless the matching percentage is very low even for the "top matched" user, which would be discouraging). Most importantly, in this matching list we would expect existing friends to place "low", i.e., to appear among the top matching choices. Therefore, we can study whether our approach outperforms other similarity computation methods in placing existing friends higher on the list, as existing friends should have similar interests [5, 6]. The similarity computation method that performs better in finding actual friends would be expected to be able to outperform others in finding potential friends as well.

b. N1: the number of users in the cluster.
c. N2: the number of users in the cluster that have a network (i.e., they are connected with at least one other user, who may be in that cluster or in another one).
d. N3: the number of users in the cluster that are connected in reality as friends.
e. Average valid connections (AVC): for each user in a cluster we computed the percentage of their connections that are included in the specific cluster, and derived the average percentage.
f. Average matching (AM): This is the average matching percentage of all users of the specific cluster.
g. Average matching of connected users (AMC): This is the average matching percentage of all the connected users of the cluster.
h. Average matching of not connected users (AMnC): This is the average matching percentage of all the users of the cluster who are not connected.

We have used several additional similarity measures in our study and implemented them in eMatch in order to compare them against Egosimilar. These similarity measures include the Pearson Correlation and the Cosine Similarity [21], which were also used in [2] and were found to provide inferior results to EgoSimilar for the smaller dataset of 57 users. The other similarity measures that we used in the present work are:

1. The Jaccard Index [22], also known as the Jaccard similarity coefficient, which is a statistic used for comparing the similarity and diversity of two sample sets. The Jaccard coefficient measures similarity between finite sample sets and is defined as the size of the intersection divided by the size of the union of two sample sets, as depicted in Eq. (2) below

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} \tag{2}$$

where A and B denote the two sample sets.

2. The π coefficient [23], which is calculated as:

$$\pi = (p_o - p_e)/(1 - p_e) \tag{3}$$

where p_o is defined as the observed agreement between two raters who each classify items into a set of M mutually exclusive categories, and p_e is defined as the expected agreement by chance.

3. The κ coefficient [24], which is similar to the π coefficient and is defined again by Eq. (3). The difference between the two coefficients lies in the way the expected agreement p_e is computed. In the π coefficient, both annotators are assumed to classify items into a category via the same probability distribution, whereas the κ coefficient does not make this assumption (hence each annotator each assumed to have a different probability distribution). As explained in [25], when using the π coefficient any differences in the observed distributions of users' judgements are considered to be noise in the data. When using the κ coefficient these differences are considered to be related to the biases of the users.

By "agreement" in the case of the π and κ coefficients, and by "intersection of sets" in the case of the Jaccard index, we are referring to two users giving the same rating for a category or for an item within a category.

Finally, in order to examine the results of the above measures, users were separated into groups via the K-means clustering algorithm [26], using the matching percentages derived by each of the similarity computation approaches. The procedure will always terminate, but K-means does not necessarily find the optimal configuration. A disadvantage of K-means is its sensitivity to the random initialization of cluster centroids; generally initial centroids should be "far apart". We addressed this issue by using different centroids and computing average results over 10 independent runs. Later in the paper, in the evaluation of our new algorithm EgoSimilar+ in "Evaluation of EgoSimilar+" section, we focused on finding the appropriate number of clusters by utilizing *silhouettes* [27].

For space economy purposes and in order to focus on the most important contributions of this study, we will only present here a summary of the results of the new evaluation of EgoSimilar.

Our results were derived for the following sets of weights: $(w_1,w_2) = (0.25, 0.75)$, $(0.5, 0.5)$, $(0.75, 0.25)$ and for $w_{change} = 0.3$, a value which was shown for both the larger dataset and the smaller one in [2] to provide the overall best results across all similarity computation methods. It did not provide the best results in all cases for EgoSimilar, which often had better results for values of 0.1 or 0.2, but for fairness and uniformity purposes we are showing the results for $w_{change} = 0.3$. We should emphasize again that we are interested in $w_1 < w_2$ as our work is focused on achieving a more specific (items-oriented) matching between users than a more generic

Table 3 Average matching difference between connected and not-connected users

Similarity computation method	Difference		
	$w_1 = 0.25$ (%)	$w_1 = 0.5$ (%)	$w_1 = 0.75$ (%)
EgoSimilar	5.03	4.98	4.92
Pearson	3.45	3.72	4.01
Cosine	3.69	2.98	2.28
Jaccard index	1.96	2.19	2.36
π coefficient	1.69	1.93	2.03
κ coefficient	1.97	2.24	2.45

Table 4 Average friends' placement

Similarity computation method	Placement		
	$w_1 = 0.25$	$w_1 = 0.5$	$w_1 = 0.75$
EgoSimilar	109.92	105.74	102.44
Pearson	111.46	114.07	117.20
Cosine	119.23	119.35	119.66
Jaccard index	107.45	112.56	118.90
π coefficient	103.62	110.56	117.02
κ coefficient	101.11	108.72	114.23

(categories-based) one. However, we also experimented with the cases where $w_1 = w_2$ and $w_1 > w_2$ in order to study the behavior of the different similarity computation methods.

Our results showed that in regard to the comparison between EgoSimilar, the Pearson Correlation and the Cosine Similarity, there were no changes in the conclusions for this larger dataset when compared with the small dataset in our previous work. More specifically, EgoSimilar outperforms both methods in terms of distinguishing between already connected and not already connected users (i.e., already connected users have a higher matching percentage). In regard to the average friends' placement, EgoSimilar also continues to outperform the Cosine Similarity and the Pearson Correlation, as in [2], by placing existing friends "lower" (i.e., closer to the top) in the users' matching list. The reason that EgoSimilar excels is that both the Cosine Similarity and Pearson Correlation only examine the current ratings of each category/item, by each of the two users. EgoSimilar, however, tries to be more sophisticated by using weights to take advantage of the popularity of the rated items.

Tables 3 and 4 present the average results for all similarity computation methods for the number of clusters K taking all values between 5 and 20.

The results indicate that:

1. EgoSimilar outperforms all similarity computation methods in terms of distinguishing between already connected and not already connected users for all (w_1, w_2) weights.

2. EgoSimilar also outperforms all similarity computation methods in terms of the average friends' placement, which is the most important metric in our study, as explained above, for $(w_1,w_2) = (0.5, 0.5)$ and $(w_1,w_2) = (0.75, 0.25)$.
3. However, EgoSimilar is outperformed in terms of the average friends' placement by all similarity computation methods, except the Cosine Similarity and the Pearson Correlation, for $(w_1,w_2) = (0.25, 0.75)$. This is an important negative result, given that our focus is on placing larger importance on individual items for making friends suggestions, therefore EgoSimilar should be able to find existing friend connections by placing them "lower" in each user's list.
4. All similarity computation methods place existing friends on average at around the 36–39% mark (positions 102 to 110 out of 285 users). This means that on average existing friends are placed close to the middle of each user's list, whereas we would expect them to place near the top. Once again, this is a negative result, which in this case applies to all similarity computation methods used in our study.

We should also note that we attempted to create groups of ratings (i.e., {1–2}, {3–5}, {6–8}, {9–10}) to avoid cases where users may like or dislike an item almost equally but a small difference in their rating would cause the similarity computation to miss the common predilection. Therefore, we considered that users "agree" if they give a rating that belongs to the same set of ratings, as defined above. We found, however, that this choice improved the results of the π and κ coefficient only slightly (by about 0.6% in the results of Table 3 and by about 3–4 positions in terms of the average friends' placement shown in Table 4). Hence, in the rest of the paper we kept the standard definition of "agreement" for the π and κ coefficients.

EgoSimilar+

The results of our evaluation of EgoSimilar against all other similarity computation methods showed that the premise of EgoSimilar is promising but was not enough to help our proposed approach excel overall, and in particular in the cases which were of the most interest for our work on eMatch.

Therefore, we first focused on understanding the reasons why EgoSimilar is outperformed by the new similarity computation methods added to our study (Jaccard index, π coefficient, κ coefficient) for the case of $(w_1,w_2) = (0.25, 0.75)$. All three similarity computation methods that outperformed EgoSimilar focus on computing the exact agreement between users, whereas EgoSimilar computes distance. Therefore, the results in this part of our work seem to indicate that even though it is more rare, exact agreement in items leads to better results in identifying existing (and hence, also possible future) friendships, especially when the element of chance agreement is removed, as is the case for the π and κ coefficients. The improved results achieved by computing exact agreement can be, at least partially, attributed to the fact that some items are essentially categories in themselves, e.g., "football" (in the category Sports) or "pop" (in the category Music); this can lead more often to exact agreement between users than a more specific football-related or pop-related item.

The results in Table 4 show that the use of the κ coefficient achieves the best results among all other similarity computation methods and outperforms EgoSimilar for

$(w_1,w_2) = (0.25, 0.75)$. The distinctive feature of the κ and π coefficients in comparison to the Jaccard index is the removal of chance agreement from the observed agreement and the distinctive feature of the κ coefficient in comparison to the π coefficient is the "acceptance" of differences in user ratings as being related to the biases of the users, instead of noise.

Based on the above, we decided to create an improved version of EgoSimilar which we name EgoSimilar+. This new version incorporates the following differences with the original algorithm:

a. We added the calculation of biases into EgoSimilar+. Similarly to [28], a first-order approximation of the bias involved in a rating R_{ui} is

$$B_{ui} = \mu + B_u + B_i \tag{4}$$

where *u* represents the user, *i* represents the item. The bias involved in rating R_{ui} is denoted by B_{ui} and accounts for the user and item effects. The overall average rating is denoted by μ, while the parameters B_u, B_i, indicate the observed deviations of user *u* and item *i*, respectively. An example, from [28], on how the adding of biases works: suppose that we want a first-order estimate for user Joe's rating of the movie *Titanic*. Suppose that the average rating for all movies, μ, is 7.4/10 and that *Titanic* has a better than average rating and tends to be rated 1 star above the average. On the other hand, Joe is a critical user, who tends to rate 0.6 stars lower than the average. Thus, the estimate for *Titanic*'s rating by Joe would be $(7.4 + 1 - 0.6) = 7.8/10$.
We added biases in order to estimate the users' ratings for items that the user had not rated although the items belonged to the user's favorite categories (categories rated equally or higher than 7/10 by the user).

b. In recent literature on databases, skyline query processing has received very significant attention [29–31]. Skyline queries find within a database the set of points that are not dominated by any other point. A n-dimensional point is not dominated by another point if it is true that it is not worse than any other point in $(n-1)$ dimensions and is better in at least one dimension. We adapted the idea of choosing non-dominated objects into EgoSimilar+. More specifically, after adding biases as explained above, dividing users in clusters and calculating user matching, EgoSimilar+ creates, for each user, two sets of potential friends. Set A contains the non-dominated potential friends, who are shown in descending matching percentage order, and Set B contains the dominated potential friends, who are again shown in descending matching percentage order. In order to identify the non-dominated potential friends, we use Eq. (1) and we calculate user matching in each category and that category's items. A non-dominated potential friend of user X is one who does not have a smaller matching percentage with X than any other user in 8 interest categories and is better than all other potential friends in at least one interest category.

It should be noted that a potential friend of user X in set B may have a higher overall matching percentage than a potential friend of X in set A. This can happen if the user in set B has a high matching percentage with X in a specific category but is dominated

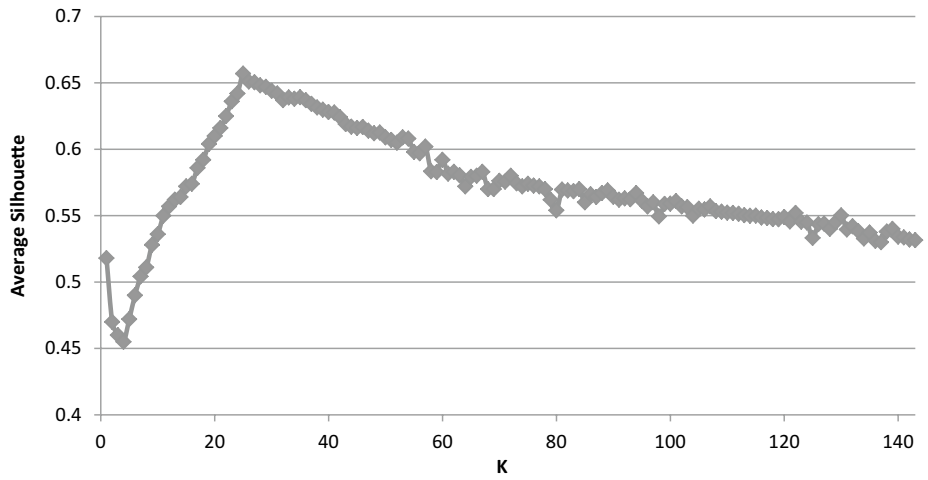

Fig. 1 Average silhouette for EgoSimilar+

in another category. Still, the fact that users in set A are not dominated lead us to place them "lower" (closer to the top) in user X's matching list.

Evaluation of EgoSimilar+

As mentioned in "Evaluation of EgoSimilar" sectiom, in order to find the appropriate number of clusters to use for K-means clustering in our dataset, we utilized *silhouettes* [27]. Silhouettes are a widely-used graphical aid for the interpretation and validation of cluster analysis. A silhouette shows which objects lie well within their cluster and which ones are merely somewhere in between clusters. If we take any user i of a cluster A we define as a(i) the average dissimilarity of i to all other objects of A, and as b(i) the dissimilarity of i to all objects of the second-best cluster then the silhouette s(i) is computed as:

$$s(i) = (b(i) - a(i))/\max(a(i),\ b(i)) \tag{5}$$

By "dissimilarity" in our study we are referring to the Euclidean distance between user vectors.

Equation (5) indicates that the best possible clustering (i.e., s(i) being close to 1) is achieved when the "within" dissimilarity is much smaller than the smallest "between" dissimilarity b(i). In this case user i is well-clustered. When s(i) is close to zero, this means that a(i) and b(i) are approximately equal and hence it is not clear to which of the two clusters user i should be assigned. When s(i) is close to -1, this means that the clustering is erroneous as user i is closer to the second best cluster and should have been assigned to it.

Silhouettes are especially useful because they can help identify cases where we have set k to be too low or too high; in both cases s(i) would be low, in the first due to a high a(i) and in the second due to a low b(i).

To find the appropriate K we studied the 286 users of our datasets and we clustered them with K ranging from 1 to 143, i.e., up to the case where we would have on average two users in each cluster.

Figure 1 shows the average silhouette for all objects (users) in the dataset, for different values of K (average over 10 independent runs for each value).

Our results are qualitatively similar with those in [32] where for a different problem it was again shown through silhouettes that the increase of K up to a point increases the probability of a user being in the best possible cluster, however the general quality of the solution decreases with a too big increase of K. We derived the best silhouette when K is around 25, i.e., for an average number of 11–12 users per cluster. The best K values when using all other similarity computation methods in eMatch were in the range of [20, 23], and for each method we used its best K for the results that follow, in order to make a fair comparison.

In order to make a fair comparison between EgoSimilar+ and the other similarity computation methods we initially implemented on them the same new ideas that we implemented on EgoSimilar+, which are presented in "EgoSimilar+" section. However, the first idea, of adding biases, leads to a smaller exact agreement between users and this in turn led to worse results for the κ coefficient, π coefficient and Jaccard index. The addition of biases had a negligible effect on the Pearson Correlation and Cosine Similarity results. Therefore, for fairness reasons we implemented for all five similarity computation methods only the second idea, of finding and presenting first the non-dominated potential friends.

Figure 2 presents the matching difference between connected and not-connected users for "the best" version of all similarity computation methods (best K, addition of the new idea or ideas the improve the method's results). EgoSimilar+ is shown not only to excel, once again but to significantly improve its results in comparison to EgoSimilar. Especially for the case that is of the most interest for us, i.e., $(w_1,w_2)=(0.25, 0.75)$, EgoSimilar+ shows a 34% improvement in comparison to EgoSimilar in distinguishing between already connected and not already connected users.

The actual matching percentages between users in each cluster vary for all similarity computation methods between 50 and 70%, with the exception of the Cosine Similarity metric which, when used in eMatch, shows an average matching percentage

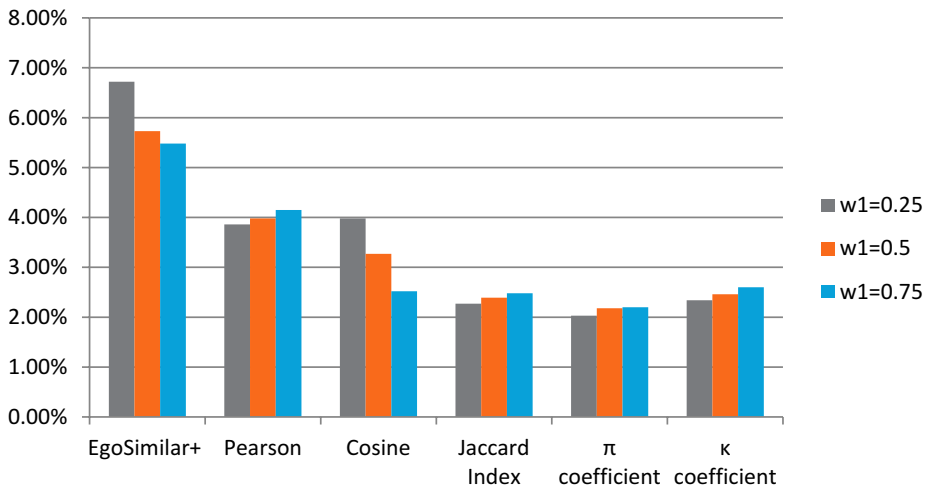

Fig. 2 Average matching difference between connected and not-connected users

larger than 80% between the users in most clusters. However, the actual matching percentage is of little value. The only substantial effect that it might have, especially in the case of not connected users, is that a quantitatively higher percentage might be more intriguing for a user in order to decide to communicate with another user. What is truly substantial is the order in which "matching users" appear on the user's screen, in decreasing percentages (high to low), where, as it will be explained below, EgoSimilar+ clearly outperforms all similarity computation metrics.

Table 5 presents the average friends' placement results for EgoSimilar+ and the other similarity computation methods, again all of them in their "best" version. It is clear from the results presented in the Table that:

1. EgoSimilar+ now outperforms all similarity computation methods in terms of the average friends' placement, which is the most important metric in our study, for all values of (w_1,w_2), including (w_1,w_2) = (0.25, 0.75) which is the most important case of our study, as explained earlier.
2. The improvement of EgoSimilar+ through the use of the two new ideas (adding biases, finding and promoting non-dominated potential friends) is very substantial, leading it to place existing friends on average at around the 29% mark (position 83/285). This improves our confidence on the quality of friend recommendations that EgoSimilar can make. EgoSimilar+ also clearly excels against all other similarity computation methods for weights (w_1,w_2) = (0.5, 0.5) and (w_1,w_2) = (0.75, 0.25), however its improvement over EgoSimilar is not as large since the critical factor in its improvement is the addition of biases to compute unknown ratings for items, and in the above cases item similarity has a smaller weight than in the (w_1,w_2) = (0.25, 0.75) case.

Figure 3 furthers shows visually the percentage improvement provided by EgoSimilar+ in terms of the average friends' placement in comparison to all other similarity computation measures. For (w_1,w_2) = (0.25, 0.75) this improvement ranges between 8.5% and 25.5% and even for (w_1,w_2) = (0.75, 0.25) the smallest improvement is still 4.5%.

Table 6 presents the same type of results as Table 5, with the difference that the same value of K for the K-means clustering was used for all the experiments, instead of using the best K for each method. The conclusions derived by Fig. 2 and Table 5 stand, once again, for the results of Table 6, which show that EgoSimilar+

Table 5 Average friends' placement for the best version of all methods

Similarity computation method	Placement		
	$w_1 = 0.25$	$w_1 = 0.5$	$w_1 = 0.75$
EgoSimilar+	82.41	92.36	100.80
Pearson	99.85	102.44	108.71
Cosine	110.56	110.91	111.32
Jaccard index	93.26	99.23	109.02
π coefficient	92.43	99.59	107.38
κ coefficient	90.05	98.40	105.57

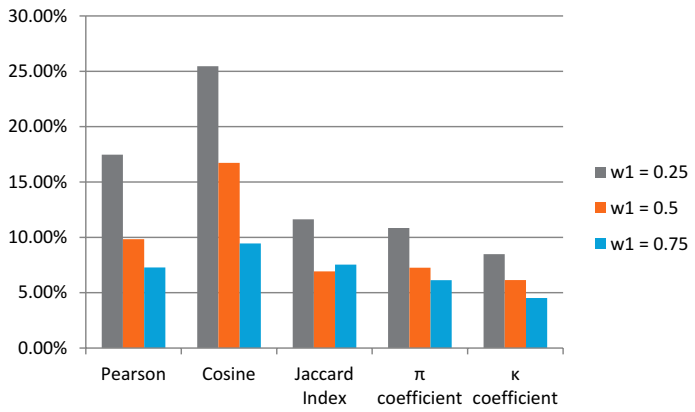

Fig. 3 Percentage of improvement offered by EgoSimilar+ in average friends' placement in comparison to all other similarity computation measures

Table 6 Average friends' placement for K = 40

Similarity computation method	Placement		
	$w_1 = 0.25$	$w_1 = 0.5$	$w_1 = 0.75$
EgoSimilar+	84.35	95.08	102.09
Pearson	102.47	106.26	112.10
Cosine	114.32	114.75	114.99
Jaccard index	95.41	102.89	111.53
π coefficient	93.70	100.82	109.41
κ coefficient	91.52	99.62	107.48

outperforms all other similarity computation method in reard to the average friends' placement in the users' matching list.

Conclusions and future work

We have presented and proposed a user similarity computation algorithm, Egosimilar+, with the aim of using it to find and connect people with common interests in the same geographical area. The algorithm is incorporated into a mobile application that serves as a "friend" recommendation system. EgoSimilar+ adapts ideas and techniques from the recommender systems literature and the skyline queries literature and combines them with our own ideas on the importance and utilization of item popularity. Our proposed algorithm is compared against five well-known similarity computation methods from the literature and is shown to excel in comparison with all of them, improving by 4.5–25.5% their results in terms of identifying true friends based on their interests.

The idea for eMatch, and hence the need for an algorithm like EgoSimilar+, was created by the fact that the contemporary way of life leads a large number of people to spend much time away from home, often alone among strangers. Therefore, it makes sense for them to connect right on the spot with someone close by who shares their interests. This is a decision that can be made quickly with the help of an intelligent

application, as opposed to decisions regarding finding possible life partners, which would usually require much more thought and study from the user (other applications focus on this area). Even at home, however, users spend a large amount of time using their mobile devices. Therefore, even users who want to take their time with evaluating possible friends will have the opportunity to do so.

One limitation of the existing work is the fact that the extended dataset is still relatively small. In future work, we will use EgoSimilar+ in large datasets from other sources in order to provide recommendations for users/items and we will compare it once again against benchmark similarity computation methods. We also intend to incorporate semantic similarity computation algorithms into eMatch, to further improve the clustering and the implicit (via the matching percentage) friendship recommendations. The use of such algorithms is important, so that relevant concepts, names and items will be linked automatically by the application (e.g., soccer and football, or soccer and Manchester United). The incorporation of spell check software is also important, in order to avoid spelling errors that can cause the algorithm to miss a commonly liked or disliked item by two users.

Authors' contributions
GS analysed the extended dataset, produced and analyzed the results of the evaluation of EgoSimilar. PK collected the extended dataset and analyzed the results of the evaluation of EgoSimilar. PK also designed and evaluated EgoSimilar+. Both authors read and approved the final manuscript.

Author details
[1] School of Electrical and Computer Engineering, Technical University of Crete, Chania, Greece. [2] School of Engineering and Information Technology, Murdoch University, Science and Computing Building 245, SC1.012, 90 South Street, Murdoch, WA 6150, Australia.

Acknowledgements
The authors wish to sincerely thank the developer of eMatch, Georgia Athanasopoulou, for her valuable help during the time that this work was conducted.

Competing interests
The authors declare that they have no competing interests.

Availability of data and materials
The datasets used and analysed in this study are available from the corresponding author on reasonable request.

Funding
This was not a funded research project.

Publisher's Note

Received: 25 September 2018 Accepted: 26 November 2018
Published online: 12 December 2018

References

1. Oommen BJ, Yazidi A, Granmo O-C (2012) An adaptive approach to learning the preferences of users in a social network using weak estimators. J Inf Process Syst 8:191–212
2. Athanasopoulou G, Koutsakis P (2015) eMatch: an android application for finding friends in your location. Mob Inf Syst J. Article ID 463791
3. Athanasopoulou G (2013) https://androidappsapk.co/detail-ematch-com-tuc-ematch/Accessed. 06 Nov 2018
4. Farrahi K, Zia K (2017) Trust reality-mining: evidencing the role of friendship for trust diffusion. HumanCentric Comput Inf Sci 7:4
5. Duck SW, Craig G (1978) Personality similarity and the development of friendship: a longitudinal study. Br J Soc Clin Psychol 17:237–242
6. Werner C, Parmelee P (1979) Similarity of activity preferences among friends: those who play together stay together. Soc Psychol Quart 42:62–66
7. Han X, Wang L, Crespi N, Park S, Cuevas A (2015) Alike people, alike interests? Inferring interest similarity in online social networks. Decis Support Syst 69:92–106

8. Lee D (2015) Personalizing information using users' online social networks: a case study of CiteULike. J Inf Process Syst 11:1–21
9. Souri A, Hosseinpour S, Rahmani AM (2018) Personality classification based on profiles of social networks' users and the five-factor model of personality. HumanCentric Comput Inf Sci 8:24
10. Youhoo (2018) http://appcrawlr.com/android/youhoo. Accessed 06 Nov 2018
11. GeoSocials (2018) http://appcrawlr.com/android/geosocials. Accessed 06 Nov 2018
12. Jiveocity (2018) http://appcrawlr.com/android/jiveocity. Accessed 06 Nov 2018
13. Liben-Nowell D, Kleinberg J (2007) The link prediction problem for social networks. J Assoc Inf Sci Technol 58:1019–1031
14. Hannon J, Bennett M, Smyth B (2010) Recommending Twitter users to follow using content and collaborative filtering approaches. In: Paper presented at the 4th ACM conference on recommender systems (RecSys), Barcelona; 2010
15. Wan S et al (2013) Informational friend recommendation in social media. In: Paper presented at the 36th international ACM SIGIR conference on research and development in information retrieval (SIGIR), Dublin; 2013
16. Han X et al (2016) CSD: a multi-user similarity metric for community recommendation in online social networks. Expert Syst Appl 53:14–26
17. Diez J, del Coz JJ, Luaces O, Bahamonde A (2008) Clustering people according to their preference criteria. Expert Syst Appl 34:1274–1284
18. Gan M, Jiang R (2013) Constructing a user similarity network to remove adverse influence of popular objects for personalized recommendation. Expert Syst Appl 40:4044–4053
19. Hwang D, Kim D (2017) Nearest neighbor based prototype classification preserving class regions. J Inf Process Syst 13:1345–1357
20. Wu J et al (2017) Weighted local Naïve Bayes link prediction. J Inf Process Syst 13:914–927
21. Mekouar L, Iraqi Y, Boutaba R (2012) An analysis of peer similarity for recommendations in P2P systems. Multimedia Tools Appl 60:277–303
22. Jaccard P (1908) Nouvelles Recherches Sur la Distribution Florale. Bulletin de la Societe Vaudoise des Sciences Naturelles 44:223–270
23. Scott WA (1955) Reliability of content analysis: the case of nominal scale coding. Public Opin Quart 19:321–325
24. Cohen J (1960) A Coefficient of agreement for nominal scales. Educ Psychol Measur 20:37–46
25. Di Eugenio B, Glass M (2004) The kappa statistic: a second look. Comput Linguistics 30:95–101
26. Forgy EW (1965) Cluster analysis of multivariate data: efficiency versus interpretability of classifications. Biometrics 21:768–769
27. Rousseeuw PJ (1987) Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. J Comput Appl Math 20:53–65
28. Koren Y, Bell R, Volinsky C (2009) Matrix factorization techniques for recommender systems. Computer 42:42–49
29. Borzsony S, Kossman D, Stocker K (2001) The skyline operator. In: Paper presented at the 17th international conference on data engineering (ICDE), Heidelberg; 2001
30. Papadias D, Tao Y, Fu G, Seeger B (2005) Progressive skyline computation in database systems. ACM Trans Database Syst 30:41–82
31. Zhang K et al (2017) Probabilistic skyline on incomplete data. In: Paper presented at the 26th ACM international conference on information and knowledge management (CIKM), Singapore; 2017
32. Thuillier E, Moalic L, Lamrous S, Caminada A (2018) Clustering weekly patterns of human mobility through mobile phone data. IEEE Trans Mob Comput 17:817–830