Future Generation Computer Systems 78 (2018) 440–450

Contents lists available at ScienceDirect

Future Generation Computer Systems

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

Query personalization using social network information and collaborative filtering techniques

Dionisis Margaris [a], Costas Vassilakis [b,*], Panagiotis Georgiadis [a]

[a] Department of Informatics and Telecommunications, University of Athens, Greece

[b] Department of Informatics and Telecommunications, University of the Peloponnese, Akadimaikou G. K. Vlachou, 22100, Greece

HIGHLIGHTS

- A social network-based query personalization algorithm is presented.
- The algorithm considers influencers from the user's social network.
- Both the user's and the influencers' preferences are used for personalizing queries.
- The algorithm is evaluated both performance- and quality-wise.

ARTICLE INFO

Article history:
Received 20 June 2016
Received in revised form
26 December 2016
Accepted 8 March 2017
Available online 10 March 2017

Keywords:
Social networks
Personalization
Collaborative search
Database query transformation
Presentation of retrieval results

ABSTRACT

Query personalization has emerged as a means to handle the issue of information volume growth, aiming to tailor query answer results to match the goals and interests of each user. Query personalization dynamically enhances queries, based on information regarding user preferences or other contextual information; typically enhancements relate to incorporation of conditions that filter out results that are deemed of low value to the user and/or ordering results so that data of high value are presented first. In the domain of personalization, social network information can prove valuable; users' social networks profiles, including their interests, influence from social friends, etc. can be exploited to personalize queries. In this paper, we present a query personalization algorithm, which employs collaborative filtering techniques and takes into account influence factors between social network users, leading to personalized results that are better-targeted to the user.

1. Introduction

The enormous growth of available content in current applications, and web applications in particular, has created information overload for the application users, necessitating the use of personalization techniques to alleviate this problem [1], by prioritizing or limiting information presented to the users according to its perceived value for them. With the advent of social networks, such as Facebook [2] and Twitter [3], used by millions of people every day, large volumes of data generated by these networks are widely available, and researchers seek methods to exploit these data for personalization purposes [4–8]; these data are deemed of high value in the context of personalization, because of the importance and the intrinsic relationship with people's everyday lives. However, due to the high volume of social network-generated data, identifying the data relevant to each individual user that are highly useful to support personalized queries, still remains a challenge.

A widely-used approach for making recommendations, stemming from user behavior and actions is collaborative filtering. Collaborative filtering (CF) synthesizes the informed opinions of humans (notably these opinions in many cases encompass the aspect of satisfaction), to make personalized and accurate predictions and recommendations. The biggest advantage of collaborative filtering is that explicit content description is not required (as in content-based systems): instead, traditional collaborative filtering relies only on opinions expressed by users on items either explicitly (e.g. a user enters a rating for the item) or implicitly (e.g. a user purchases an item, or clicks an advertisement banner, which indicates a positive assessment). In the context of collaborating filtering, personalization is achieved by considering ratings of "similar users", under the collaborative filtering's fundamental assumption that if users X and Y have similar behaviors (e.g., buying, watching, listening, rating assignment) on some items, they will act similarly

* Corresponding author.
E-mail address: costas@uop.gr (C. Vassilakis).

http://dx.doi.org/10.1016/j.future.2017.03.015
0167-739X/

on other items [9]. Traditional recommender systems though, assume that users are independent and ignore the social interactions among them, hence they fail to incorporate important aspects that denote interaction, tie strength and influence among users, which can substantially enhance recommendation quality [10,7]. Social network data-based recommender systems take into account static data from the user profile, such as location, age or gender, complemented with dynamic aspects stemming from the user behavior and/or social network state such as user preferences, item's general acceptance, and influence from social friends [10,7]. Towards this direction, the metric of *tie strength* between users of social networks has been modeled, quantifying the projected influence of one user to another. This metric can be exploited to further enhance the choice of recommenders, so as to consider the opinions and choices of users that have a high influence on the user for which the recommendation is generated [4,11,12].

Query personalization is an important area in personalization, in which queries are dynamically enhanced with related preferences stored in a user profile with the purpose of providing more focused answers [13]. Query personalization systems may be used either by end-users who directly pose the queries that are subsequently personalized, or as an underlying infrastructure in any database-supported application, in order to tailor the results returned by queries to the application user's profile. To this end, query personalization systems have been studied in a number of works [13–17]. These works consider user personal and/or collaborative preferences that are stored in a preference repository, they do exploit however opportunities offered by social media data to enhance query adaptation quality.

In this paper, we propose a novel algorithm for query personalization, based on social network information. The algorithm exploits both the users' choices of items (browsing and rating) and the influence information from social networks to suitably adapt queries for each user. Query adaptation is performed via (re)writing the query sorting specification to order the qualifying data according to their projected interest for the user. The proposed algorithm is evaluated both in terms of personalization accuracy and execution performance.

The proposed algorithm extends the state of the art query personalization algorithms by (a) exploiting social graphs for improving personalization quality and (b) arranging for mapping the outcome of the personalization procedure proposing an efficient query rewriting technique.

The rest of the paper is structured as follows: Section 2 presents the proposed algorithm. An evaluation of the algorithm, both in terms of personalization quality and performance is given in Section 3. Section 4 overviews related work, while finally, Section 5 concludes the paper and outlines future work.

2. The query personalization algorithm

In this section, we first describe the concept of influence in social networks, which is central to the proposed algorithm. Subsequently we elaborate on the representation of the information model used by the algorithm: this model includes user preferences, the influence information from social networks and a sample database schema for exemplifying the functionality of the personalization. Finally, we detail the operation of the algorithm and its phases.

2.1. Influence in social networks

Within a social network, "social friends" greatly vary regarding the nature of the relationship holding among them: they may be friends or strangers, with little or nothing in between [18]. Users have friends they consider very close, and know each other in real life, acquaintances they rarely meet or communicate in real life, and persons they have never actually met with, such as singers, actors and athletes. According to Anagnostopoulos et al. [19], three main causes of correlation in social networks exist: *influence* (also known as induction), where the action of a user is triggered by one of her friend's recent actions (e.g. when a user buys a product because one of her friends has recently bought the same product); *homophily*, which means that individuals often establish "social friendship" with others who are similar to them, and hence perform similar actions (e.g. sharing a common interest, such as mountaineering); and *environment* (also referenced as *external influence*), where external factors are correlated both with the event that two individuals become friends and also with their actions (e.g. two inhabitants of the same city posting pictures of the same landmarks in an online photo sharing system can become "social friends").

Bakshy et al. [5] suggest that a social network user responds significantly better to recommendations that originate from friends of the social network to which the user has high *tie strength*; the strength of the directed tie between users i and j is computed as:

$$W_{i,j} = \frac{C_{i,j}}{C_i}$$

where C_i is the total number of communications posted in a certain time period in the social network by user i, whereas $C_{i,j}$ is the total number of communications posted by user i on the social network during the same period and are directed at user j or on posts by user j. In [5], a period of 90 days is considered for computing the tie strength. Note that a post may be directed towards multiple users (individually specified recipients or all members of a group, such as "close friends"), in which case it contributes to the tie strength of all its recipients. Because of this fact, it holds that

$$\sum_j W_{i,j} \geq 1.$$

2.2. The information repository

In order to be able to personalize a user's queries, the information repository must include (a) data concerning the items that users want to retrieve through their queries (b) data regarding the influencers for the particular user, which are computed from the social network graph and (c) the preferences on items for both the user and her influencers. Regarding the items that users want to retrieve through their queries, in this work we consider a typical database schema used in query personalization research (e.g. [14, 16]), representing movies, actors and directors. With respect to the users' preferences on items, we consider a simple model representing preference of users towards movies. This kind of information can be extracted from social networks [20–22] or from specialized site databases (e.g. IMDB [23]) provided that linkage between the social network profiles and specialized site accounts are established by the users. Finally, regarding the influencer information, we consider periodic snapshots computed from the social network graph. The complete information repository schema is illustrated in Fig. 1. The left side of this figure depicts the schema extension used in our work to support the personalization process, which includes the following information:

- *Users*: users are extracted from the social network API or from dataset files (e.g. the MovieLens dataset [24] or the Amazon datasets [25,26]). Since our personalization scheme does not include demographic or other user-profile based information, only the user id is retained in the database.

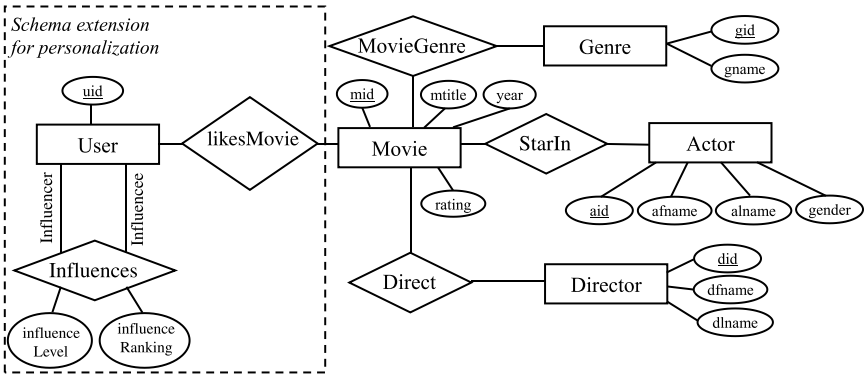


Fig. 1. Extended movies database ER model.

- *Movie likings*: this relationship represents which users like which movies. This information can be extracted from the social network API, e.g. from Facebook's graph API [27], or from dataset files. In some cases, the data source (API or dataset) does not provide assertions that a user likes particular movies, but rather movie ratings (e.g. the MovieLens dataset [24] or the Amazon datasets [25,26]), while Facebook provides movie ratings [28] complementary to the movie liking assertions. When movie ratings are available, we consider that the user likes movies that she has rated with a value equal to or greater than the median of all the ratings she has provided.
- *Influences*: this relationship represents the influencers of each user. The *influence level* between two users is an arithmetic quantity corresponding to the tie strength described in Section 2.1 above. The *influence ranking* attribute reflects the relative ranking among the influencers of a particular user: for any user U, her influencer having the highest influence level on her is assigned an influence ranking value of 1, the influencer with the second highest influence level is assigned a value of 2 and so forth.

Note that the adoption of a simple preferences model, recording only users' preference towards movies is used in this work to simplify the presentation of the personalization algorithm. However, the query personalization algorithm can accommodate more complex preferences models, such as those described in [16]. Firstly, it is straightforward to record preferences towards other elements of the data that users want to retrieve, such as genres, actors and directors, or relationships among data e.g. movies of specific genres that a particular actor appears in. Secondly, the algorithm presented in [16] initially computes the results considering individual preferences to generate partial results and then combines partial results to generate the final query result; therefore, within an environment that multiple preferences are considered, each individual preference can be processed for personalization as presented below in Section 2.3, to produce partial results, and finally all partial results can be combined using either the EXCLUDE_COMBINE or the REPLICATE_DIFFUSE algorithms described in [16] to produce the final query result.

2.3. Query personalization

Having the information listed in Section 2.2 available, the personalization algorithm may commence its operation. The operation of the algorithm is divided in three phases: in phase 1 it fetches the information from the social network; in phase 2 it exploits this information to realize query personalization; and, finally, in phase 3 it updates the information retrieved in phase 1. For performance reasons, steps 1 and 3 are performed in an offline fashion, while phase 2 is executed online. These algorithm phases are described in the following paragraphs.

Phase 1—offline initialization. In this phase, the system populates the *User*, *likesMovie* and *Influences* relations, using information from the social network. The *User* and *likesMovie* relations are directly copied from the social network graph, corresponding to user nodes and "user like" arcs directed towards movie nodes, respectively. Regarding the *Influences* relation, for each user u the recommendation algorithm uses the formula presented in Section 2.1 to compute the tie strength between user u and each user u' in the social neighborhood of u. Afterwards, the social network friends that are considered as *prominent influencers* are maintained and stored in the user profile. The criteria for determining the prominent influencers are as follows:

- for each user we consider up to $N = 30$ influencers
- we only maintain influencers having an influence level $\geq$0.18.

These values have been determined experimentally; the relevant experiment is described in Section 3.1.

Phase 2—online operation: Once the algorithm has been bootstrapped, its online execution phase, in order to perform query personalization and execution, may commence. This phase of the algorithm is executed each time a user submits an SQL query, and exploits information regarding (a) the likings of the user submitting the query and (b) the likings of her influencers, in order to provide more focused answers. The operation of the algorithm is presented in detail in the following paragraphs.

1. The top N influencers $\{i_i, i_2, \ldots, i_N\}$ for the active user (AU) and the influence level of each one are extracted from the information repository. Since the *Influences* relationship extension, as populated in phase 1, contains only the top N influencers for each user, it suffices to extract *all* influencers for AU:

```
SELECT influencer, influenceLevel
FROM Influences
WHERE Influences.influencee = AU
```

2. For each influencer i_j, the algorithm computes a partial result, corresponding to the recommendation based on the particular influencer's movie likes, as these are recorded in the *likesMovie* table. This is performed by finding the movies that satisfy the criteria specified in the original query, and then assigning to each movie an influencer-based score as follows:

$$inflScore_{AU}(m, i) = \begin{cases} InfluenceLevel(i, AU), \\ \quad if\ influencer\ i\ likes\ movie\ m \\ 0, \quad otherwise. \end{cases}$$

Effectively, the partial result computation query is formulated as:

```
PartialResult_i: SELECT  IQ.*, if(isnull(likesMovie.mid), 0, influenceLevel_i) score
                 FROM    (InitialQuery) IQ LEFT JOIN likesMovie
                             ON IQ.mid = likesMovie.mid
                 WHERE   likesMovie.uid = i_i;
```

where IQ is the initial query and function *isnull* checks if its argument has a null value.[1] These partial results are then merged to produce an aggregate result corresponding to the recommendation based on the influencers' likings; in this result, each movie is assigned a score equal to the sum of all individual influencer-based scores:

```
PartialResult_INFL: SELECT <initial select list>, sum(score) as influencerScore
                    FROM   (SELECT * FROM PartialResult_1 UNION ALL
                            SELECT * FROM PartialResult_2 UNION ALL
                            …
                            SELECT * FROM PartialResult_N)
                    GROUP BY <initial select list>
```

3. Similarly to step (2) which considers the influencers' preferences, the algorithm also considers the preferences of the active user. In this step, each movie is assigned a user preference based score as follows:

$$userScore_{AU}(m) = \begin{cases} 1, & \text{if the active user AU likes movie } m \\ 0, & \text{otherwise.} \end{cases}$$

To construct a partial result corresponding to the active user's preferences, the following query is formulated:

```
PartialResult_AU:  SELECT IQ.*, if(isnull(likesMovie.mid), 0, 1) userScore
                   FROM (InitialQuery) IQ LEFT JOIN likesMovie
                        ON IQ.mid = likesMovie.mid
                   WHERE likesMovie.uid = AU.id;
```

4. Finally, the algorithm combines the partial results computed in steps 2 and 3 above to produce the final query result. In this combination, the results are ordered by their perceived interest for the user that has issued the query. The perceived interest is quantified by employing the simple additive weighting method for multiple criteria decision [29], considering (a) the influencer-based score computed in step 2 and (b) the user preference-based score computed in step 3. According to the simple additive weighting method, each criterion is assigned a weight, and the weighted sum of the scores assigned to each criterion is computed for every evaluated item as the item's final score. In our algorithm, the weight assigned to the influencer-based score will be denoted as W_{infl}, with $0 \leq W_{\text{infl}} \leq 1$, while the weight assigned to the user preference-based score is set to $(1 - W_{\text{infl}})$. The value of W_{infl} has been determined experimentally, and the relevant experiment is discussed in Section 3.2.
The query combining the results can be written as:

```
FullResult:  SELECT <initial select list>
             FROM   PartialResult_INFL LEFT JOIN PartialResult_AU
                    ON PartialResult_INFL.mid = PartialResult_AU.mid
             ORDER BY W_infl * influencerScore + (1 - W_infl) * userScore;
```

The above transformation is not efficient for execution, since it needs to compute the initial query once for each partial result (or store it in a temporary table and use it from there), while additionally each partial result query (including the $\text{PartialResult}_{\text{AU}}$ query) performs a join with the *likesMovie* table to implement the calculation of scores. An alternative and more efficient method to compute the same final result would be to compute the influencer-based score (InflScore) and the user-based score (UScore) for each movie and then join these score relations with the result of the initial query IQ. Considering the datasets used in [30,31], the average number of movie reviews per user is 2.2, with a maximum of 2654; therefore in all cases, the whole bulk of the InflScore and UScore relations can be maintained in-memory, and be efficiently joined with the result of IQ using a hash-based join method [32]. Taking these into account, the personalized query can be expressed as a single SQL query as follows[2]:

```
FullResult:  SELECT <initial select list>
             FROM   (IQ) iq
                    LEFT JOIN (
                        SELECT likesMovie.mid, sum(influenceLevel) influencerScore
                        FROM likesMovie JOIN influences
                             ON likesMovie.uid = influences.influencer
                        WHERE influences.influencee = AU
                        GROUP BY likesMovie.mid) InflScore
                        ON iq.mid = InflScore.mid
                    LEFT JOIN (
                        SELECT likesMovie.mid, 1 as userScore
                        FROM likesMovie
                        WHERE likesMovie.uid = AU) UScore
                        ON iq.mid = UScore.mid
             ORDER BY W_infl * ifnull(influencerScore, 0) + (1 - W_infl) * ifnull(userScore, 0);
```

Note that the arrangement presented above includes, in the final result, all movies in the result of the initial query IQ, regardless of whether these are liked by the active user or any of her influencers; movies that are not liked by any considered user, will have total score of 0 (the total score being the value computed in the ORDER BY clause), hence they will appear last in the result set.

Phase 3–repository update. Since both the contents of the social networks (users and communications with users) and the users' likings on movies are dynamic, the corresponding information elements of our model need to be updated, to maintain their consistency with the current status of the social network information. The updates that need to be performed are as follows:

(i) After a movie has been liked by a user, a relative record needs to be inserted in the *likesMovie* table.
(ii) Periodically, the influencers and their rankings are recomputed, since the overall number of communications of each user and the number of targeted communications between an ordered pair of users may change.
(iii) Finally, new users can be accommodated in the social network or users may depart; again, the personalization schema may be periodically updated to reflect these changes.

3. Experimental evaluation

In this section, we report on our experiments aiming:

(a) to determine the values of parameters that are used in the algorithm; in particular, this set of experiments is targeted to identify the optimal value for parameter N, corresponding to the number of influencers that must be maintained per user, so as to offer useful and effective recommendations and for parameter W_{inf}, corresponding to the weight assigned to the influencer-based recommendations.
(b) to evaluate the performance of the proposed approach, both in terms of execution time (the time needed to make the personalized ordering) and users' satisfaction regarding the offered recommendations.

For our experiments we used a DELL PowerEdge M910 blade server equipped with four Intel Xeon E7−4830 @ 2.13 GHz CPUs, 256 GB of RAM and one 900 GB disk with a transfer rate of 200 MBps, which hosted the H2 database (H2 v. 1.4.193 [33]), storing the tables corresponding to the database schema depicted Fig. 1, including the users' profiles (users and preferences) and the influence data retrieved from the social network.

In order to determine the value for parameter W_{infl} we conducted a user survey in which 25 people participated. The participants were students and staff from the University of Athens community, coming from the computer science department. 16 of the participants were men and 9 were women, with their ages ranging between 20 and 53 years, with a mean of 31. All users were frequent Facebook users, using this social network at least three times a week (accounting for 22 h of weekly online presence, with a minimum of 8 h) and being members for 19 months or more; the

[1] http://www.h2database.com/html/functions.html#ifnull.

[2] The ifnull function in this query returns its first argument if its value is non-null; otherwise it returns its second argument (http://www.h2database.com/html/functions.html#ifnull).

number of Facebook friends for the participants varied from 73 to 552, with a mean of 217.

To assess personalization quality, we conducted a user survey in which 50 people participated. The participants were students and staff from the University of Athens community, coming from the computer science department. 28 of the participants were men and 22 were women, and their ages range between 19 and 44 years, with a mean of 29. Regarding the participants' profile and behavior within Facebook, the minimum number of Facebook friends among the participants was 82 and the maximum was 403, with a mean of 205. All participants used Facebook regularly (at least three times a week) and had been members for at least two years.

In both experiments, for each person, we computed the relevant tie strengths with all of her Facebook friends in an offline fashion [21,22]. No user participated in both experiments.

3.1. Determining the number of influencers

The first experiment aimed to determine the number of influencers N that must be maintained by the system per user, in order to produce accurate recommendations. Recall that for each query, we seek to take into account the opinion of the strongest influencers when generating personalized results ordering; thus in this experiment, we vary the number of strongest influencers considered, seeking the point at which adding more influencers does not alter the recommendations generated. The recommendations are expected to converge when the number of influencers considered increases, because strongest influencers are added first to the influencers set, hence increments beyond some point will only contribute with influencers having only a weak level of influence, and thus not being capable of altering the personalized outcome.

To find the value of parameter N after which recommendations converge, we developed 100 sets of 1000 synthetic users (unfortunately, no relevant dataset, including past user selections, as well as tie strengths was available) to generate recommendations for. The distributions of the tie strengths in each generated dataset followed the one reported in [34]. Subsequently, we generated the top 10 relevant recommendations for different values of N varying from 5 to 40, and finally we calculated the probability that a recommendation generated for a user considering his/her N strongest influencers (recom@N) is different than the corresponding recommendation generated considering his/her $N + 1$ strongest influencers (recom@$N + 1$). The results, depicted in Fig. 2, show that when the 30 strongest influencers in the category are considered, any further increment in the number of influencers considered will only marginally modify the recommendations generated (only 2% of the top-10 recommendations are modified—a manifestation of the *diminishing returns* law [35]), hence in the subsequent experiments we will consider only the 30 strongest influencers for each user.

Another issue to consider is whether a threshold should be applied to the influence level of an influencer, so as to be included in the *prominent influencers* set of a user. According to [36], approximately 30% of the tie strengths in Facebook have values less than 0.1, while an additional 17% has values less than 0.2. Such tie strengths are expected to contribute only marginally to the outcome of the recommendation, since in the algorithm presented in Section 2 movie likings are weighed by the influence level (i.e. the tie strength) of the corresponding influencers. Moreover, Ferrara et al. [34] report that for users with a small number of ties, strong ties outnumber weak ties: this suggests that even for users with a small overall number of friends, at least some of these friends will have high values of tie strength, thus their likings will dominate the outcome of the collaborative filtering scoring procedure. In this respect, it is worth investigating

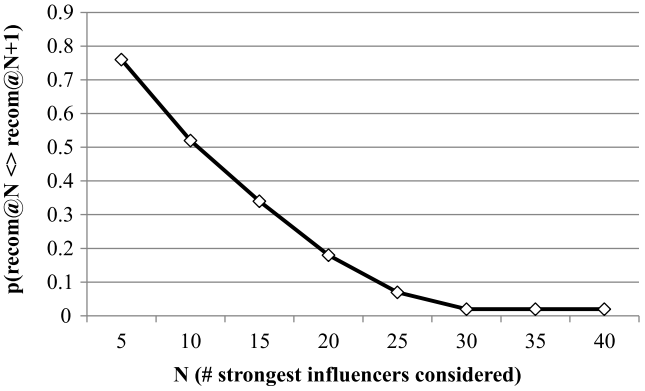


Fig. 2. Different recommendations made, due to the fact of considering 1 more recommender.

whether the removal of friends having a very low influence level (i.e. an influence level below a certain threshold) would affect the recommendation process: note that pruning each user's set of influencers according to a threshold will reduce the cardinality of the *Influences* relationship shown in Fig. 1, and is therefore expected to reduce the execution time of the queries listed in Section 2.3.

To determine the optimal value of the influence level threshold IL_{th} we conducted a second experiment, using the same dataset with the previous experiment, setting the maximum number of influencers N to 30, according to the results of the previous experiment, and varying the value of IL_{th} to gain insight on how the setting of IL_{th} affects the outcome of the recommendation process. In particular, we considered values of IL_{th} in the range of [0, 0.4] and for every value il_{th} we calculated the probability that the recommendation using the pruned influencer sets $recom@il_{th}$ is different than the recommendation using the full sets of influencers *recom@full* which is identical to applying pruning with a zero value for IL_{th}, i.e. $recom@il_0$ (again, the 10 top ranked items in the recommendations are considered in the comparison). The results of this experiment are depicted in Fig. 3. From this figure we can determine that pruning at a tie strength level up to 0.1 has practically no effect in the outcome of the recommendation process, with $p(recom@il_{0.1} <> recom@full) = 0.007$. When IL_{th} is set to the value of 0.18%, 2.1% of the tested recommendations have been found to deviate from the respective ones produced when the unpruned set of influencers is used, with 87% of these changes being either reorderings of items at positions 8–10 or substitution of items initially ranked at positions 9–10 by items initially ranked at positions 11–12, and only 3.6% of these modifications affecting the composition or the order of the five top-ranked items within the recommendation. At a practical level, such changes are insignificant to the users, as a number of studies assert that the click-through rate of the five first search results is seven to nine times higher than the click-through rate of all other results [37–39]. Increasing the influence level threshold IL_{th} beyond the point of 0.2 has been found to distort a higher ratio of the recommendations, with a higher probability that the composition or order of the five top-ranked items within the recommendation changes.

Taking the above results into account, we set the maximum number of influencers considered N to 30 and the influence level threshold IL_{th} to 0.18. Under these settings, the cardinality of the *Influences* relationship shown in Fig. 1 was reduced by 10.1%. This reduction is analyzed as follows:

- the influencer set of users having more than 65 friends (which corresponds to approximately 62% of the overall Facebook user base [40]) remained practically unaltered

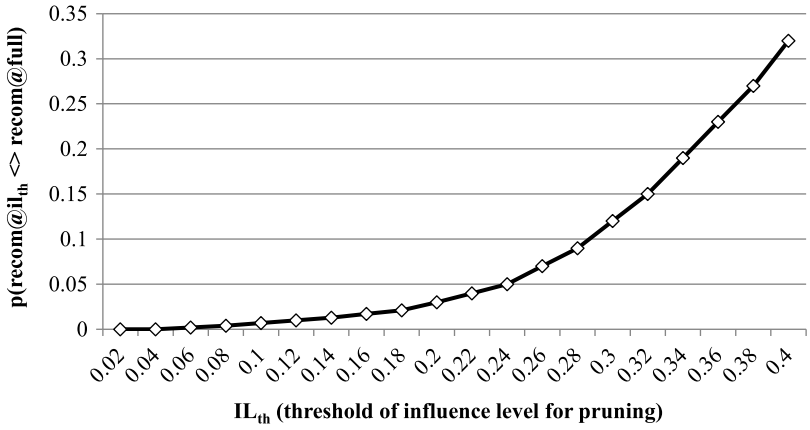


Fig. 3. Different recommendations made, due to the fact of pruning the influencer set based on an influence level threshold.

- for users with up to 30 friends (approximately 22% of the overall Facebook user base [40]) the average reduction in the cardinality of the influencers' set was 33.1% (recall that for users with small numbers of friends, the probability that a tie is strong is higher [34], hence the reduction achieved is less than 42.6% of the average number of ties having strength less than the threshold of 0.18) and
- for users having between 30 and 65 friends, (approximately 16% of the overall Facebook user base [40]), the average reduction in the cardinality of the influencers' set was 19.6%. While in this case the probability that a tie is weak is higher, the elimination of friends with weak ties from the influencer set of a user is bound to not decrease the cardinality of the influencer set below 30, especially for users whose number of friends is close to 65 (i.e. the upper bound).

3.2. Setting the weight for the influencer-based score

As discussed in Section 2.3, the proposed algorithm employs the simple additive weighting for multiple criteria decision to combine the user-based score and the influencer-based score. The actual values used for the weights of the influencer-based score (W_{infl}) and the user-based score ($1 - W_{\text{infl}}$) may significantly affect the overall performance of the algorithm: setting W_{infl} to 0 effectively reduces the algorithm to considering only the user preferences and disregarding the collaborative filtering aspect, while setting W_{infl} to 1 eliminates any effect of the user preferences, resulting to a pure collaborative filtering-based algorithm. In order to determine the optimal value for parameter W_{infl} we conducted an initial experiment in which each 25 users were invited to submit five queries each. Each query was personalized multiple times, using the values 0.1, 0.2, 0.3, 0.4, 0.5, 0.6 and 0.7 for the W_{infl} parameter. The respective queries were executed to collect the personalized results and duplicate results were merged (some user queries retrieved only a few items from the database, and in these cases some differently parametrized personalizations produced identically ranked items; in order to avoid asking users to rate the same query result multiple times – one for each personalization that produced it – the query result was presented once and the ranking was associated with all W_{infl} values that produced the particular result). Subsequently, the order of the results was randomized to avoid bias and finally users were asked to rate the quality of each result assigning to each a score on a scale of 1 (totally unsatisfactory) to 10 (totally satisfactory).

Users were asked to think aloud when assessing the results, and their comments allowed us to gain insight on the rationale behind the assigned ratings.

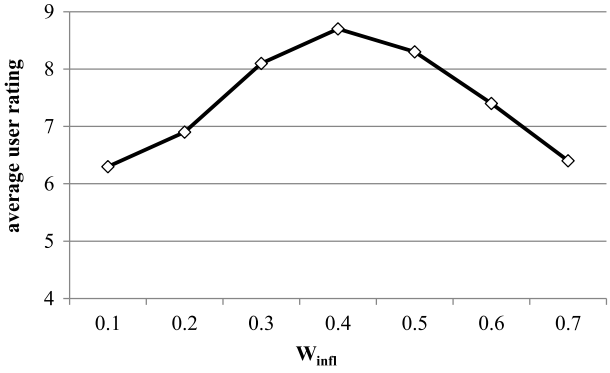


Fig. 4. Average user rating of query personalization results, with varying values for W_{infl}.

Fig. 4 depicts the results from this experiment. From these results, we can identify that the optimal value for W_{infl} is 0.4. When the value of W_{infl} was set to low values, the result of the personalization process was dominated by the own preferences of the active user, effectively considering only the results of step (3) of the second phase of the algorithm described in Section 2.3. Especially for users for whom the *likesMovie* relation was highly populated, this domination drastically reduced the serendipity dimension of the recommendations [41,42] and this was negatively perceived by subjects, particularly when the goal of the query was to find a recommendation for a new movie to watch ("This is not useful, I have already watched these movies", "Tell me something I don't already know").

When the value of W_{infl} was set to higher values, the influencer-based results, corresponding to the results of step (2) of the second phase of the algorithm described in Section 2.3, appeared to determine to a great extent the personalization process. Due to this fact, when some movie preferences of the user were not shared by her influencers, the respective movies appeared in low positions in the result list and this was commented negatively by the experiment subjects ("This is a classic one, it should have been ranked among the first five", "Why has this movie sunk down here?"). In some cases, users did not scroll down enough to see some movie they expected to be listed among the results (see also the following discussion on this issue) and complained that "movie X should be listed among the results".

It is worth noting that – in order to formulate their ratings – users in most cases reviewed up to 10 results, which is equal to the number of results that were directly visible on the screen without scrolling. The average number of results reviewed was 6.1. This is consistent with the results of other user studies [37–39], according to which the click-through rate of the five first search results is

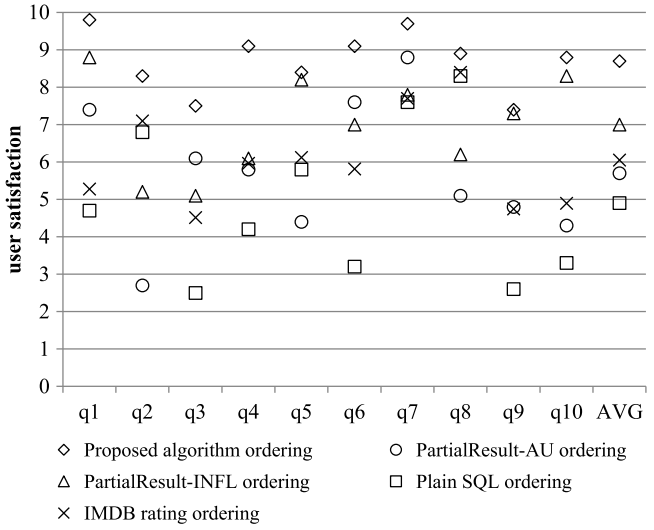


Fig. 5. Users' satisfaction of non-personalized query results vs. personalized query results.

seven to nine times higher than the click-through rate of all other results. These facts reinforce the view that ordering of results is of particularly high importance.

The results of this experiment assert that combining the results of steps (2) and (3) of the second phase of the algorithm described in Section 2.3 increases the quality of the final recommendation. Based on these results, in the subsequent experiments the value of W_{infl} is set to 0.4.

3.3. User satisfaction

In the first part of this experiment, each participant formulated her own queries against the movie database (5 queries per person) and each query was submitted to the database five times as follows:

(a) with no personalization (ordering was determined by the database system)
(b) using the personalization algorithm (the outcome of the *FullResult* query) described in Section 2,
(c) using only the part of the personalization algorithm that considers the active user's preferences (the outcome of the $PartialResult_{AU}$ query),
(d) using only the part of the personalization algorithm that considers the influencers' opinions (the outcome of the $PartialResult_{INFL}$ query),
(e) using the IMDB score for each movie. The rationale behind using the IMDB score was that movies with higher IMDB scores (a) are deemed of better quality and therefore may constitute better recommendations in the case that the query goal is to find a movie to watch and (b) are more well-known, hence it would be more probable for users to be searching for them when trying to retrieve information for a specific movie.

Subsequently the order of the results was randomized and users were asked to rate each result on a scale of 1 (totally unsatisfactory) to 10 (totally satisfactory).

Fig. 5 illustrates the results of this experiment. Queries Q1–Q10 account for the query types that users formulated most frequently within the experiment; example types are given in Table 1. The score presented in Fig. 5 for each query type is the average score of all queries of the same type submitted during the experiment.

The last column in Fig. 5 depicts the average score of all queries submitted to the system. On average, the proposed ordering algorithm (*FullResult*) is found to increase user satisfaction by 77.1% (user satisfaction average equals to 8.7 for the personalized results versus 4.9 of the non-personalized ones), while the corresponding performance edge against the IMDB rating-based ranking is 43.7%. The performance of the proposed ordering algorithm (*FullResult*) is found to be superior to that of the $PartialResult_{AU}$ variant by 52.6% (the $PartialResult_{AU}$ variant has a user satisfaction average equal to 5.7); the $PartialResult_{INF}$ exhibits a user satisfaction average equal to 7.0, being inferior to the proposed ordering algorithm by 24.3%.

The proposed algorithm also outperforms the "flat profile" approach reported in [16], which improves user satisfaction by 52.9%, and is comparable to the results of [15], which uses an elaborate preference selection model (in contrast to the simplistic preference model used in this work), showing that the potential of using social network-generated data in query personalization is high. The "net profile" approach described in [16] has the highest reported user satisfaction improvement insofar, approximately equal to 150%: this is due to the highly elaborate preferences selection model used therein while additionally, in the user experiment reported in this work, non-personalized query results have been given a very low score (an average of 3.4), having thus a substantial improvement margin (recall that in the experiment reported in our work, non-personalized query results were rated with an average score of 4.9, allowing for a maximum improvement ratio of 104%).

The smallest improvements were identified in Q8 (Movies of a specific genre in which an actor starred and released in a particular year). This is due to the fact that this query type typically yields up to two results, hence the improvement margin is very small.

At the second part of this experiment, we asked the subjects to pick, for each query they submitted, one single movie among the query results, designating it as the movie that most accurately corresponds to their query goal. The goal of this experiment part was to assess the extent to which each algorithm succeeds in placing the movies matching the users' interests in the first positions of the result list. This is of high importance considering the results of [37–39] presented above, according to which users typically examine only the first few items within a search result. Table 2 presents some example queries and the relevant results, while Fig. 6 illustrates the results from the second part of the experiment. In this figure, we may notice that the proposed algorithm places the user-designated movie first in the list in almost half the cases (47.1%) and second in about 30% of the cases; overall, the user-designated movie is ranked within the first three results in 90% of the cases. The $PartialResult_{AU}$ variant places the designated movie first in the list in approximately one out of three cases (31.3%) and second in one out of four cases (24.5%); overall, the user-designated movie is ranked within the first three results in 77.7% of the cases. The $PartialResult_{\text{INFL}}$ variant places the designated movie first in the list in 41.3% of the cases and second in 27.2% of the cases; overall, the user-designated movie is ranked within the first three results in 84.3% of the cases. The IMDB-based rating places the user-designated movie first in the list in one out of three cases (34.3%) and second in about 30% of the cases (28.50%); overall, the user-designated movie is ranked within the first three results in 82% of the cases. Finally, the plain SQL ordering places the user-designated movie first in one out of five cases (18.2%), while second position placements are approximately equal in numbers (19.4%); overall, the user-designated movie is ranked within the first three results in approximately half the cases (53.5)%.

It has to be noted here that some queries (e.g. movies that an actor starred in and were released in a particular year) yield only a few results (typically less than 3); this explains why the plain SQL ordering (effectively a random one) achieves such high frequencies in placing the subject-designated movie in the first three places.

Table 1
Sample frequent query types.

Query type id	Query type	Example query
Q1	Movies of a specific period in which an actor stared in	Rowan Atkinson (actor) movie prior to 2010
Q3	Movies of a specific genre in which an actor starred and was released in a particular period	2006–2014 drama with Robert De Niro (actor)
Q6	Movies of a specific period directed by a particular director	Steven Spielberg directed this movie in the 90s
Q8	Movies of a specific genre in which an actor starred and was released in a particular year	2007 drama with Meryl Streep (actress)

Table 2
Examples of placements of user-designated movies in the result lists.

Example query	Query type	Results of example query
Scott Adkins (actor) movie prior to 2008	Movies of a specific period in which an actor stared in (Q1)	The participant designated "Undisputed II"; the proposed algorithm placed it first in the result list, while the $Partial_{INFL}$ and IMDB orderings placed it second. $Partial_{AU}$ placed it fourth and the plain SQL query ordered it fifth.
2009–2014 comedy with Sacha Baron Cohen (actor)"	Movies of a specific genre in which an actor starred and released in a particular period (Q3)	The participant designated "The Dictator"; the proposed algorithm and the $Partial_{AU}$ variant placed it second in the result list. The plain SQL query and the $Partial_{INFL}$ variant ordered it fourth and according to the IMDB rating it was placed third.
Adrian Lyne directed this movie in the 80s	Movies of a specific period directed by a particular director (Q6)	The participant designated "Nine 1/2 weeks"; the proposed algorithm and the $Partial_{AU}$ variant placed it first in the result list while the plain SQL and $Partial_{INFL}$ orderings placed it third. According to the IMDB rating it was placed fourth.
2014–2015 comedy with Melissa McCarthy (actress)	Movies of a specific genre in which an actor starred and released in a particular year (Q3)	The participant designated "Spy"; the proposed algorithm placed it first out of 3 movies, so did the plain SQL and $Partial_{INFL}$ orderings. According to the IMDB rating and the $Partial_{AU}$ variants it was placed second.

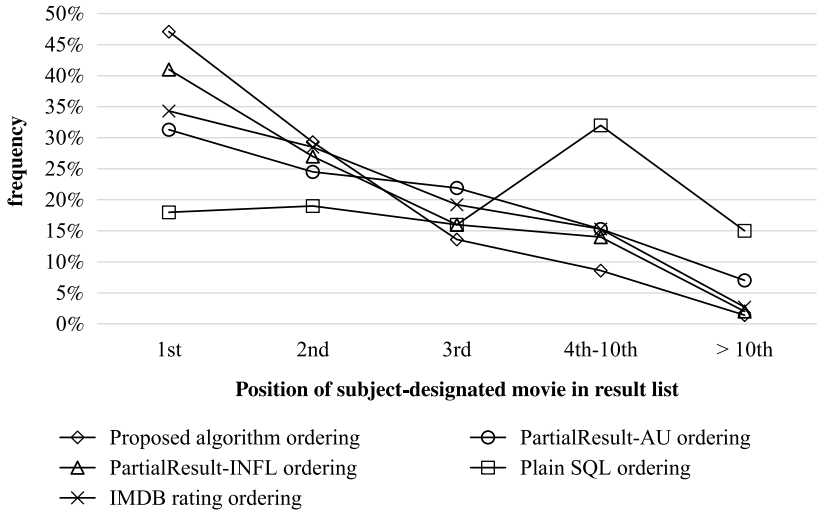


Fig. 6. Frequency of positions of the subject-designated movie within the query result.

3.4. Recommendation formulation time

The next experiment aimed to quantify the time needed to formulate the appropriate results orderings, when users execute queries on a real-world large database. We used the IMDB database [23], starting from a 10% fraction of its size (400,000 movies and analogously small numbers of directors, actors, etc.) and gradually increasing it to its full size, accounting for approximately 4 million titles, 3 million actors and actresses with 22.5 million credits, and 450,000 directors with 2.5 million credits, populating the relevant tables in the database (actors, directors, movies, etc.). Appropriate index structures were also created. To further explore the scalability of our proposal, we generated synthetic data to increase the movies database size up to 1.5 times the size of IMDB. Note that when comparing the current IMDB size [43] with its corresponding size one year earlier [44] we can note an overall increase by 12.5% in one year (titles increased by 14.4%; actors by 9.9% and actor credits by 12.4%; directors increased by 11.9% and director credits by 14.1%). This growth rate indicates that IMDB will increase its size by 50% in approximately 3 years, hence our performance metrics are valid for at least this period.

Data regarding the users and their influencers were synthetically generated. A total of 50,000,000 users were generated; each user had from 100 to 500 friends overall, with an average of 190 friends, following the mean value of friends in the Facebook social network [40].

All data were initially stored on the disk. After processing the social network data and deriving the top-N influencers per user, as described in Section 3.1, we explored both a disk-based and a memory-based storage option. When the size of the movies database was set to 150% the size of IMDB, the movie-related data (right part of the schema in Fig. 1) had a size of 8.2 GB (including indices), while the social network-related data (left part of the schema in Fig. 1) had a size of 221.3 GB (again, including indexes), accounting for an overall database size of 229.5 GB.

As far as the queries are concerned, we chose some of the most common ones typically used in query personalization works (films

of a specific year and genre; films starring a specific actor/actress or having been directed by a specific director, or achieving total score more than a specific threshold, etc.).

The results of this experiment are depicted in Fig. 7. We can observe that the extra time needed to execute the personalized query instead of the initial one, scales linearly with the size of data and the average extra time ranges between 33% and 47% in the disk-based storage scenario and between 20% and 45% in the memory-based storage scenario. It is worth noting that the overhead decreases as the movie-related database size increases, reaching its minimum values when the movie-related database size is equal to 150% of the size of IMDB.

Considering the scalability of the proposed approach, our performance experiments were run using a synthetic user base of 50,000,000 users; Facebook, which is the largest social network, currently has 1.7 billion active users [45]. This is 34 times more users than those used in our experiment, therefore the respective database size would be approximately 8 TB. Since the memory capacity of contemporary high-end servers has substantially increased to more than 10 TB (e.g. [46] can accommodate up to 12.2 TB of memory), such a server can host the entire database in-memory, considering the full Facebook user population. In our future work, we will investigate techniques for partitioning the database extension to multiple servers to further promote scalability and enhance performance.

4. Related work

In the domain of query personalization, Koutrika and Ioannidis [47] develop a personalization framework for database systems, based on user profiles and identify the basic architectural modules required to support it. Each atomic query condition is assigned a personal degree of interest according to a preference model, and subsequently the degree of interest of complex query conditions (i.e. query conditions that involve combinations of atomic query conditions) is computed through appropriate mechanisms. User preferences are stored in profiles, and the query personalization procedure first selects the preferences that are best suited to the current query, and then the selected preferences are integrated into the original user query. Koutrika et al. [14] elaborate on preference representation, storage in user profiles, and selection of the preferences that will be used in the query enhancement phase.

Koutrika and Ioannidis [15] present a generalized model of user preferences for database information, including positive and negative preferences, as well as hard and soft ones (i.e. preferences that must be satisfied and preferences for which it is desirable but not necessary to satisfy). In this work, preference information is stored in user profiles and a number of algorithms are introduced that exploit this information to enhance incoming queries and generate ranked, personalized answers.

Koutrika [13] defines collaborative query personalization, by introducing collaborative concepts in query personalization. In this work, the query enhancement phase takes into account both the user's personal preferences, as well as preferences of people who are deemed to be "similar" to the current user (analogously to *recommenders* in collaborative filtering).

In [16], Koutrika and Ioannidis consider multiple granularities in preferences, with more fine-grained (i.e. specific) preferences overriding the more coarse-grained (i.e. generic) ones. Preferences are modeled using a network that accommodates an inheritance relationship, and algorithms are given for (a) identifying relevant preferences, (b) modifying queries accordingly, and (c) processing these queries to obtain personalized answers.

Freitas et al. [17] address query personalization in diverse settings taking into account the user context; to this end, along with the user interests and personal information, the system maintains a set of *environment concepts* which are related to the setting where the user interacts and the application is executed, such as location, access device, connection bandwidth and type of application (e.g. e-commerce, game etc.). This work uses an ontology as a representational model and also reports on the implementation of a service that manages user context and can provide it as input to any query personalization system.

Sarwat et al. [48] present RecDB; a full-fledged database system that provides personalized recommendation to users, using the open source database system PostgreSQL, and demonstrate the effectiveness of RecDB using recommendation applications. Akbarnejad et al. [49] present a fragment-based instantiation of QueRIE, a recommender system that assists users when interacting (querying) with large relational database systems, while generating real-time personalized query recommendations, however, in terms of performance, QueRIE's recommendation engine is able to generate real-time recommendations with an average of 25 s.

None of the above mentioned query personalization algorithms takes into account the social networks' structure, as well as influencing factors between social network users, to produce personalized query results. The proposed algorithm aims to fill this gap, in order to provide more accurate query results and efficient execution of the personalized queries. The relevant works from the collaborative filtering and the social network-based recommendation systems are reviewed below.

In the domain of personalization systems, numerous approaches for formulating recommendations have been proposed in the existing literature. Collaborative filtering (CF) formulates personalized recommendations on the basis of ratings expressed by people having similar tastes with the user for which the recommendation is generated; taste similarity is computed by examining the resemblance of already entered ratings [50]. Research has proven that CF-based recommendation approach is the most successful and widely used approach for implementing recommendation systems [51]. Collaborative filtering can be further distinguished in user-based and item-based approaches [52]. In user-based CF, a set of nearest neighbors of the target user is first identified, and the prediction value of items that are unknown to the target user is then computed according to this set. On the other hand, item-based CF proceeds by finding a set of similar items that are rated by different users in some similar way. Subsequently, predictions are generated for each candidate item, for example, by taking a weighted average of the active user's item ratings on these neighbor items [53].

Recently, with the advent of social networking, social network recommendation has received considerable research attention. Konstas et al. [8] investigate the role of social networks' relationships in developing a track recommendation system based on a common collaborative filtering item recommendation method, by taking into account both the social annotation and friendships inherent in the social graph established among users, items and tags. Arazy et al. [4] outline a conceptual recommender system design within which the structure and dynamics of a social network contribute to the dimensions of trust propagation, source's reputation and tie strength between users, which are then taken into account by the system's prediction component to generate recommendations. Quijano-Sanchez et al. [12] enhance a content-based recommender system by including the trust between individuals, users' interaction and aspects of each user's personality in the recommendation algorithm.

Cai et al. [6] propose a model that captures the bilateral role of user interactions within a social network and formulate collaborative filtering methods to enable people to people recommendation. In this model, users can be similar to other

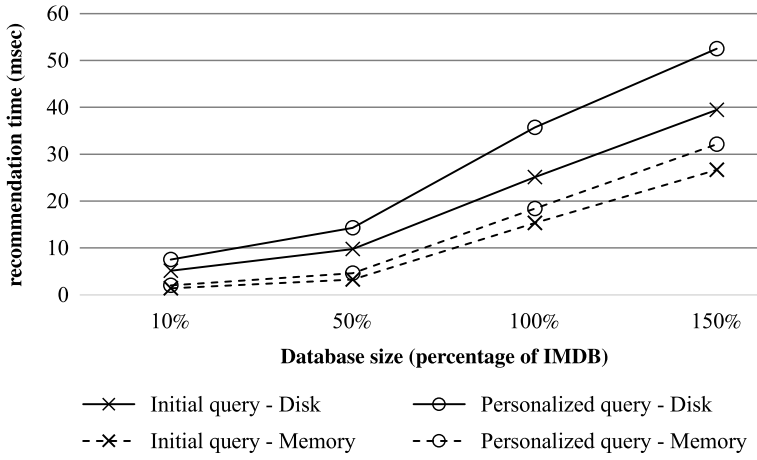


Fig. 7. Recommendation formulation time for varying degrees of database size.

users in two ways—either having similar "taste" for the users they contact, or having similar "attractiveness" for the users who contact them. A neighbor-based collaborative filtering algorithm was also developed, to predict, for a given user, other users they may like to contact, based on user similarity in terms of both attractiveness and taste. He and Chu [7] analyze data from a social network and establish that friends have a tendency to select the same items and give similar ratings; they also show that using social network data within the recommender system improves the prediction accuracy but also remedies the data sparsity and cold-start issues inherent in collaborative filtering. Eirinaki et al. [54] propose a trust-aware system for personalized user recommendations in social networks. In this work, trust (and distrust) between people is a central concept in the algorithm that assists members of a community to make decisions about other members of the same community. Suresh et al. [55] examine how social data, ratings, and reviews can be combined to create personalized rankings of reviews, tailored to each individual user, thus providing each user with efficient access to the reviews that are deemed most suitable for her.

The need for extending recommendation algorithms to take into account social networks' structure is established in Gilbert and Karahalios [18] as well as in Bakshy et al. [5]; in these works, measures for user tie strength and social influence are also proposed. In more detail, [18] presents a predictive model that maps social media data to tie strength, distinguishing between *strong* and *weak* ties with generally accepted accuracy and illustrates how modeling tie strength can improve social media design elements, including privacy controls, message routing, friend introductions and information prioritization, while Bakshy et al. [5] measure social influence via social cues on an economically relevant form of user behavior and average rates of response, demonstrate the substantial consequences of including minimal social cues in advertising and measure the positive relationship between a consumer's response and the strength of their connection with an affiliated peer. Oechslein and Hess [11] also assert that a strong tie relationship has positive influence on the value of a recommendation.

5. Conclusions and future work

In this paper we have presented an algorithm for ordering query results when social networks users execute queries, taking into account (a) the behavior of users in the past and (b) the influencing factors between social networks users. The algorithm used for performing the personalized ordering follows the collaborative filtering algorithm. The proposed algorithm contributes to the state-of-the-art by taking into account the influencing factors between social network users, on the results to be ordered. The proposed algorithm has been experimentally validated regarding (i) its performance, and (ii) the accuracy of result ordering, with respect to user interests (users' satisfaction to the ordering produced) and the results are encouraging.

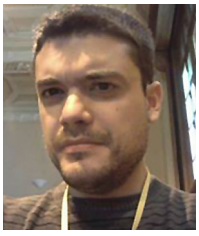
One aspect of our future work will focus on studying selected items' temporal dimension, since users' behavior, as well as interests may change through time, and its absence may lead to inaccurate recommendations. We also plan to consider in our experiments alternative methods for computing tie strength, including dimensions highlighted in recent researches such as [56]. Techniques for partitioning the database extension to multiple servers, to further promote scalability and enhance performance will also be investigated.

We finally plan to take into account description keywords ("hate that movie", "I love this actor", "best movie ever", "this director is one of the best", etc.) that follow check-ins and tags, in order to achieve more accurate recommendations in the future.

References

[1] S. Berkovsky, J. Freyne, Web personalization and recommender systems, in: Proceedings of the 21st ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2015, pp. 2307–2308.
[2] Facebook, social network home page, 2016. https://www.facebook.com (accessed 15.05.16).
[3] Twitter, twitter social network home page, 2016, https://twitter.com/ (accessed 10.05.16).
[4] O. Arazy, N. Kumar, B. Shapira, Improving social recommender systems, IT Prof. (2009).
[5] E. Bakshy, D. Eckles, R. Yan, I. Rosenn, Social influence in social advertising: Evidence from field experiments, in: Proceedings of the 13th ACM Conference on Electronic Commerce, 2012, pp. 146–161.
[6] X. Cai, M. Bain, A. Krzywicki, W. Wobcke, Y. Sok Kim, P. Compton, A. Mahidadia, Collaborative Filtering for People to People Recommendation in Social Networks, in: AI 2010: Advances in Artificial Intelligence, Lecture Notes in Computer Science, vol. 6464, 2010, pp. 476–485.
[7] J. He, W.W. Chu, A social network-based recommender system (SNRS), Ann. Inf. Syst. 12 (2010) 47–74.
[8] I. Konstas, V. Stathopoulos, J.M. Jose, On social networks and collaborative recommendation, in: Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval, 2009, pp. 195–202.
[9] C.L. Hwang, K. Yoon, Multiple Criteria Decision Making, in: Lecture Notes in Economics and Mathematical Systems, vol. 186, Springer-Verlag, 1981.
[10] Facebook, Facebook interest targeting, 2016. https://www.facebook.com/help/188888021162119 (accessed 15.05.16).
[11] O. Oechslein, T. Hess, The value of a recommendation: The role of social ties in social recommender systems, in: Proceedings of the 47th Hawaii International Conference on System Science, 2014, pp. 1864–1873.
[12] L. Quijano-Sanchez, J.A. Recio-Garcia, B. Diaz-Agudo, Group recommendation methods for social network environments, in: Proceedings of the 3rd Workshop on Recommender Systems and the Social Web within the 5th ACM International Conference on Recommender Systems, RecSys'11, 2011.
[13] G. Koutrika, Personalization of structured queries with personal and collaborative preferences, in: Multidisciplinary ECAI'06 Workshop about Advances on Preference Handling, 2006.

[14] G. Koutrika, E. Pitoura, K. Stefanidis, Preference-based query personalization, in: B. Catania, L.C. Jain (Eds.), Advanced Query Processing, in: Intelligent Systems Reference Library Series, vol. 36, 2013, pp. 57–81.
[15] G. Koutrika, Y. Ioannidis, Personalized queries under a generalized preference model, in: Proceedings of the Proceedings of the 21st International Conference on Data Engineering, ICDE'05, 2005, pp. 841–852.
[16] G. Koutrika, Y. Ioannidis, Personalizing queries based on networks of composite preferences, ACM Trans. Database Syst. (TODS) 35 (2) (2010).
[17] M. Freitas, J. Silva, D. Bandeira, A. Mendonça, D. Souza, A.C. Salgado, A user context management approach for query personalization settings, in: Proceedings of the Sixth IEEE International Conference on Semantic Computing, 2012, pp. 33–335.
[18] E. Gilbert, K. Karahalios, Predicting tie strength with social media, in: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI'09, 2009, pp. 211–220.
[19] A. Anagnostopoulos, R. Kumar, M. Mahdian, Influence and correlation in social networks, in: Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD'08, 2008, pp. 7–15.
[20] Facebook, Facebook for developers, Graph API user reference User – Movies, 2016. https://developers.facebook.com/docs/graph-api/reference/user/movies/ (accessed 10.05.16).
[21] Facebook, Facebook for developers, Graph API reference, 2016. https://developers.facebook.com/docs/graph-api/reference (accessed 10.05.16).
[22] Neo4j Blog, Mashups with the Facebook Graph API and Neo4j. http://neo4j.com/blog/mashups-with-the-facebook-graph-api-and-neo4j/ (accessed 23.12.16).
[23] IMDB Database Statistics. http://www.imdb.com/stats (accessed 22.12.16).
[24] MovieLens datasets. http://grouplens.org/datasets/movielens/ (accessed 28.11.16).
[25] J.J. McAuley, R. Pandey, J. Leskovec, Inferring Networks of Substitutable and Complementary Products, in: Proceedings of KDD 2015, pp. 785–794.
[26] J. McAuley, C. Targett, J. Shi, A. van den Hengel, Image-based recommendations on styles and substitutes, in: Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR, 2015.
[27] Facebook, Facebook Graph API User Movies documentation. https://developers.facebook.com/docs/graph-api/reference/user/movies/ (accessed 28.11.16).
[28] Facebook, Facebook Graph API Video Rates documentation, https://developers.facebook.com/docs/reference/opengraph/action-type/video.rates (accessed 28.11.16).
[29] I. Kaliszewski, D. Podkopaev, Simple additive weighting—A metamodel for multiple criteria decision analysis methods, Expert Syst. Appl. 54 (2016) 155–161.
[30] J. McAuley, C. Targett, J. Shi, A. van den Hengel, Image-based recommendations on styles and substitutes, in: Proceedings of the SIGIR 2015 Conference.
[31] J. McAuley, R. Pandey, J. Leskovec, Inferring networks of substitutable and complementary products, in: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, August 2015, pp. 785–794.
[32] P. Mishra, M.E. Eich, Join processing in relational databases, ACM Comput. Surv. 24 (1) (1992) 63–113.
[33] H2, The H2 database, 2016. http://www.h2database.com/html/main.html (accessed 10.12.16).
[34] E. Ferrara, P. De Meo, G. Fiumara, A. Provetti, The role of strong and weak ties in Facebook: a community structure perspective. http://giacomofiumara.altervista.org/pubblicazioni/ur03.pdf (accessed 28.11.16).
[35] A.P. Samuelson, W.D. Nordhaus, Microeconomics, seventyth ed., McGraw-Hill, ISBN: 0071180664, 2001.
[36] V. Arnaboldi, A. Guazzini, A. Passarella, Egocentric online social networks: Analysis of key features and prediction of tie strength in facebook, Comput. Commun. 36 (10–11) (2013) 1130–1144.
[37] L.A. Granka, T. Joachims, G. Gay, Eye-tracking analysis of user behavior in WWW search, in: Proceedings of SIGIR 2004, Sheffield, South Yorkshire, UK, July 25–29, 2004.
[38] Chitica, The Value of Google Result Positioning, 2013. available at https://chitika.com/google-positioning-value (accessed 17.06.16).
[39] P. Petrescu, Google Organic Click-Through Rates in 2014, 2014. available at https://moz.com/blog/google-organic-click-through-rates-in-2014 (accessed 17.06.16).
[40] Facebook, Facebook anatomy, 2016. https://www.facebook.com/notes/facebook-data-team/anatomy-of-facebook/10150388519243859 (accessed 28.11.16).
[41] M. Ge, C. Delgado-Battenfeld, D. Jannach, Beyond accuracy: Evaluating recommender systems by coverage and serendipity, in: Proceedings of RecSys'10, September 26–30, 2010, pp. 257–260.
[42] M. de Gemmis, P. Lops, G. Semeraro, C. Musto, An investigation on the serendipity problem in recommender systems, Inf. Process. Manag. 51 (5) (2015) 695–717.
[43] IMDB, IMDB statistics. http://www.imdb.com/stats (accessed 16.12.16).
[44] IMDB, IMDB statistics (current for November 7, 2015). https://web.archive.org/web/20151107032414/http://www.imdb.com/stats (accessed 16.12.16).
[45] Statista, Number of Facebook users worldwide 2008–2016, https://www.statista.com/statistics/264810/number-of-monthly-active-facebook-users-worldwide/ (accessed 10.12.16).
[46] DELL, PowerEdge R930 Rack Server specs. http://www.dell.com/us/business/p/poweredge-r930/pd?ref=PD_OC (accessed 10.12.16).
[47] G. Koutrika, Y. Ioannidis, Personalization of queries in database systems, in: Proceedings of the 20th International Conference on Data Engineering, ICDE'04, 2004.
[48] M. Sarwat, J. Avery, M. Mokbel, RecDB in action: recommendation made easy in relational databases, in: Proceedings of the VLDB Endowment, VLDB Endowment Homepage Archive, vol. 6, no. 12, August 2013, pp. 1242–1245.
[49] J. Akbarnejad, G. Chatzopoulou, M. Eirinaki, S. Koshy, S. Mittal, D. On, N. Polyzotis, J. Swarubini, V. Varman, SQL QueRIE Recommendations, PVLDB 3 (2) (2010) 1597–1600.
[50] J.B. Schafer, D. Frankowski, J. Herlocker, S. Sen, Collaborative filtering recommender systems, in: The Adaptive Web, in: Lecture Notes in Computer Science, vol. 4321, 2007, pp. 291–324.
[51] W. Zhang, T. Chen, J. Wang, Y. Yu, Optimizing top-n collaborative filtering via dynamic negative item sampling, in: Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR'13, 2013, pp. 785–788.
[52] J.L. Herlocker, J.A. Konstan, L.G. Terveen, J.T. Riedl, Evaluating collaborative filtering recommender systems, ACM Trans. Inf. Syst. 22 (1) (2004) 5–53.
[53] M. Balabanovic, Y. Shoham, Fab: content-based, collaborative recommendation, Commun. ACM 40 (3) (1997) 66–72.
[54] M. Eirinaki, M.D. Louta, I. Varlamis, A trust-aware system for personalized user recommendations in social networks, IEEE Trans. Syst. Man Cybern.: Syst. 44 (4) (2014) 409–421.
[55] V. Suresh, S. Roohi, M. Eirinaki, I. Varlamis, Using social data for personalizing review rankings, in: Proceedings of the 6th Workshop on Recommender Systems and the Social Web, RSWeb 2014.
[56] A. Sinan, D. Walker, Tie strength, embeddedness and social influence: A large-scale networked experiment, Manage. Sci. 60 (6) (2014) 1352–1370.



Dionisis Margaris is a Postdoc Researcher in the Department of Informatics and Telecommunications of the University of Athens, in the area of Information Systems. He has received his degree, his master's degree and his Ph.D. from the Department of Informatics and Telecommunications at the University of Athens in 2007, 2010 and 2014, respectively. He has published 12 papers in international journals and conferences and his research interests include service-oriented architectures, databases, social networks and personalization.

Costas Vassilakis is an Associate Professor in the Department of Informatics and Telecommunications of the University of the Peloponnese, in the area of Information Systems. He has received his degree from the Department of Informatics, University of Athens in 1990 and his Ph.D. from the same department in 1995. During the period 1991–1995 he was receiving a scholarship from the Greek State Scholarship Foundation. His work includes research in systems software and analysis and design of software systems, and has published more than 140 relevant papers in international journals and conferences. He has also participated in more than 25 international and national projects, mainly involving information and software systems. His research interests include information systems, semantic web, real-time and dependable systems, service-oriented architectures as well as information presentation aspects.

Panagiotis Georgiadis is an Emeritus Professor in the Department of Informatics and Telecommunications of the University of Athens, Greece. He holds a B.Sc. in Physics, an M.Sc. and a Ph.D. in Computer Science. He has been a regular member of the Senate of University of Athens (1992–1994), Director of the Computer Systems & Applications Division of the Department of Informatics (1994–1996) and Secretary General for Information Systems by the Greek Ministry of Finance (20/4/1997–16/1/2002). His research interests include Distributed Systems, Simulation and Management of Information Systems. He has authored more than 55 scientific articles in international journals and conferences, and contributed in national and European research projects.