Manuscript Details

Manuscript number	IS_2018_396_R1
Title	Efficiently Using Contextual Influence to Recommend New Items to Ephemeral Groups
Article type	Research paper

Abstract

Group recommender systems suggest items to groups of users that want to utilize those items together. These systems can support several activities that can be performed together with other people and are typically social, like watching TV or going to the restaurant. In this paper we study ephemeral groups, i.e., groups constituted by users who are together for the first time, and for which therefore there is no history of past group activities. Recent works have studied ephemeral group recommendations proposing techniques that learn complex models of users and items. These techniques, however, are not appropriate to recommend items that are new in the system, while we propose a method able to deal with new items too. Specifically, our technique determines the preference of a group for a given item by combining the individual preferences of the group members on the basis of their contextual influence, the contextual influence representing the ability of an individual, in a given situation, to guide the group's decision. Moreover, while many works on recommendations do not consider the problem of efficiently producing recommendation lists at runtime, in this paper we speed up the recommendation process by applying techniques conceived for the top-K query processing problem. Finally, we present extensive experiments, evaluating: (i) the accuracy of the recommendations, using a real TV dataset containing a log of viewings performed by real groups, and (ii) the efficiency of the online recommendation task, exploiting also a bigger partially synthetic dataset.

Keywords	group recommendations; influence; context; efficiency

Corresponding Author	Emanuele Rabosio
Corresponding Author's Institution	Politecnico di Milano
Order of Authors	Elisa Quintarelli, Emanuele Rabosio, LETIZIA TANCA

Suggested reviewers	Ludovico Boratto, Sihem Amer-Yahia, Georgia Koutrika

Submission Files Included in this PDF

File Name [File Type]

answers.pdf [Response to Reviewers]

group_rec_infsyst.pdf [Manuscript File]

Submission Files Not Included in this PDF

File Name [File Type]

highlights.txt [Highlights]

abstract.txt [Abstract]

QuintarelliRT - LatexSourceFiles.zip [LaTeX Source File]

To view all the submission files, including those not included in the PDF, click on the manuscript title on your EVISE Homepage, then click 'Download zip file'.

ANSWERS TO THE REVIEWER'S COMMENTS

First of all we would like to thank the Reviewers and the Editor for their valuable work and comments on the manuscript, which hopefully helped us enhance it in this revised version. In the following, we answer the Reviewers' observations point by point, also specifying the modifications that have been applied to the paper. Moreover, all the modifications have been underlined in red in the new version of the paper.

However, before that we need to raise an important point, on which we need clarification about the Editor's intention: in this new version of the paper we have not added a user study, because our impression is that it would add very little to the results of our experiments. Indeed, as explained both in the paper and in point 9 below (please check it), we believe that, when controlling our results against the test set extracted from the real, very large dataset we use, the accuracy measure is a very reasonable esteem of the user satisfaction.

On the other hand, preparing a user study would require much time: we should collect tastes of individual users, then group them to be able to compute the influence of users, and then we should aggregate users in new (ephemeral) groups to evaluate if they appreciate our recommendations.

This said, if the Editor considers essential the user study, we would like to ask for an extension of the deadline for the review because it would require much time, since we need to interact for multiple sessions with our students (they would be the users for the experimental session).

ID	Reviewer's comment	Answer
1	Section 2 (Related Work) requires a better structure organized with either subsections or bullets. In its current form, it is hard to distinguish the categories and transitions between.	**We have now organized the related work in subsections, as follows: 1) Persistent Groups, 2) Ephemeral Groups, 3) Influence in Group Recommendation, 4) Efficiency of Group Recommenders.**
2	There should be either a dedicated "data model section" or a "notation reference table" to ensure the fluidity in Section 3.	**We have now added two new subsections: "3.1 - Context Model" and "3.2 - Data Model and Problem Statement". These subsections explain in a more organized way and in greater detail the context and data models.**

(*Continued on next page*)

continued from previous page

ID	Reviewer's comment	Answer
3	Although not explicitly mentioned, the problem is positioned in a cold-start setting: recommending to groups with no history. However, three following elements are introduced as inputs to the group recommendation algorithm: "log of past individual choices", "item descriptions", and "log of the past group choices". How do authors justify this paradox?	**Our work is positioned in the item cold start setting, i.e., our technique is able to recommend items that are new in the system. When a new item enters the system it does not have a history of past interactions with the users, but comes with its *item description* in terms of features (e.g., the genre of a movie) that can be leveraged by a content-based recommendation system to include the new item in the recommendation lists. In fact, as explained in Section 3.3.1, we suppose that the individual preferences of the users are computed through a content-based recommender.** **On the contrary, this paper does not tackle the user cold start problem: our users are not supposed to be new in the system, so they are associated with a *log of past individual choices*.** **Moreover, the proposed approach targets the general scenario of ephemeral groups, which are ad-hoc groups, potentially constituted by users that have never been together before. Please notice that this does not mean that the users are new in the system, but just that they may be together in the same group for the first time. The *log of the past group choices* contains the history of the choices performed by groups of users in the past. However, since the groups are ephemeral, we cannot suppose that the target group, for which the recommendations are generated, has already appeared in the past. The users that are in that group may have been part of other groups in the past (and this information is useful to compute the user's influence), but we suppose that the target group may have a composition never seen before.** **We have now added to the introduction a footnote underlining that the users composing the ephemeral groups are not new in the system, a brief sentence clarifying that we set ourselves in the item cold start setting, and a paragraph better explaining the ephemeral group scenario.**
4	Despite its importance in formulating the recommendation problem, the notion of "context" has never been introduced formally. It remains unclear what properties are associated to a context. Examples of "daytime" and "night" help the reader the get the intuition, but a more formal definition is needed.	**We have introduced a new subsection "3.1 - Context Model", in which we have explained more formally the notion of context and described the way in which we have employed it in this paper.**

(*Continued on next page*)

continued from previous page

ID	Reviewer's comment	Answer
5	Following the previous comment on the lack of structure for contexts, I am wondering if the authors have looked at the relation between contexts? For instance, the "night" context comes after the "daytime" context in a temporal sequence. Hence it is intuitive to consider the impact of the former on the latter.	**As we have now better clarified in the new subsection "3.1 - Context Model", this work adopts a simple and plain representation of the contexts, seen as sets of dimension/value pairs; the only hierarchical aspect envisaged in the representation is the presence of the universal context $\tilde{c}$. We have introduced the context with the aim of dividing the log in portions that, as it has actually been confirmed by the experiments, are associated with different behaviors of the users, which translate in different values of preferences and influence. Our aim, therefore, has been to keep contexts like `daytime` and `night` separated, in order to obtain more accurate recommendations. Nevertheless, more complex models might compute preferences and influence of user u in context c exploiting also the activities performed by u in contexts other than c, using different weights; these weights could be based on some form of similarity between contexts, or learned from data. As we have now added to the conclusion, we leave a study of this kind as an interesting future work.**
6	Computation of "individual preferences" and "influences" is considered out of the loop. However, it is appreciated to discuss their impact on the top-k group recommendation step. For that, the authors may briefly discuss the space and time complexities of the offline approaches by citing the related state of the art.	**We have now extended Section 3.3.1 with a brief discussion of two suitable approaches for the computation of individual preferences, commenting also time and space complexities. Moreover, in Section 3.3.2 we have now reported the time and space complexities of the influence computation through Equation (2).**
7	Two characteristics of groups are studied in the experiments: "size" and "heterogeneity". What about demographic attributes of group members? For instance, male and female groups can be compared.	**We have extended the experimental evaluation considering also subsets of the test items based on demographic features of the group members, in particular gender and age slot. About the gender, experiments have been conducted on the groups containing only females, only males, and only children aged between 0 and 14 (whose gender is not specified in our data). Regarding the age slots, we have tested the following ones: 15-30, 31-50, more than 50. These new subsets of the test items are detailed in Section 5.2.3, while the tables of Section 5.2.4 have been extended with the related results. The discussion of Section 5.2.5 has also been extended commenting the new results; in particular, a new subparagraph "Results when Varying the Group Demographic Features" has been added to the *Comparison with the Competitors from the Literature*, and the *Relevance of the Context in Influence Computation* has been complemented with further considerations.**

(*Continued on next page*)

continued from previous page

ID	Reviewer's comment	Answer
8	While there is a focus on "TV watching activities" in the manuscript, the authors may comment on a typical domain-dependent group size. This will shed some light on issues like the importance of time/space efficiency.	**We have now extended the discussion, in Section 4.2, of time and space efficiency, emphasizing the advantages that our algorithm can bring. Regarding the group size, we remark that the required space does not depend on the group size ($O(\|\mathcal{I}\|\|\mathcal{U}\|\|\mathcal{C}\|)$, see Section 4.2). On the contrary, the time complexity is affected by the group size, and we have now included it in the time complexity expression (in the original submission the group size had been considered as negligible). We have also discussed typical group sizes in different real scenarios, on the basis of our data for the TV domain and relying on what is reported in the literature for other domains. In particular, in our real TV dataset the average group size is 2.26. Then, in the literature we have found for place visit the values 3.98 and 4.68, for online events 97.64, for real events 10.75 and 20.3, and for online communities of interest 31. These numbers are actually quite small with respect to the possible numbers of items, but however it seems reasonable to assume that in certain domains, like online events and communities of interest, the size of the group may grow consistently. In such cases, the usage of an efficient algorithm like ours would become even more crucial.**

(*Continued on next page*)

continued from previous page

ID	Reviewer's comment	Answer
9	While the experiments discuss the accuracy and efficiency of the proposed approach, its usability remains unexamined. It is necessary to understand how real groups of users appreciate recommendations obtained from this approach. How do authors justify the lack of a user study?	**One of the datasets we have used in the experiments contains a large log with both individual and group viewings performed by real users, and we consider the existence of this dataset as a veritable luck for us since it enabled us to compute the influences of each user in each group they belonged to. As explained in Section 5.2, we have evaluated the quality of our proposal by checking if the item we recommended was actually the movie chosen by a *newly formed group* (i.e. a group that was in the test set but not in the training set).** **Our argument goes as follows: for a user study, we should collect the tastes of the individual users, group them to be able to compute the users' influences, and then we should aggregate users in new (ephemeral) groups to evaluate if each of them appreciates our recommendations. The only additional information that we would collect from this procedure would be the individual satisfaction of a user that, having anyway decided to make a certain choice when in the group, might say that he/she is not contented with it. We think that such an info is relatively relevant, since the aim of the recommendation is not to satisfy each single user but to guess the choice that would be made in each specific condition, thus the relevant check to be done is that finally the item recommended by our system is actually the one that was viewed.** **As a consequence, we feel quite comfortable to think that, when controlling our results against the test set, we can take the accuracy measure as a very reasonable esteem of the user satisfaction. Thus, the reason why the paper lacks a user study is that the dataset in our opinion allowed us to perform realistic experiments, that can reasonably be considered as covering also the point of view of usability.**
10	Algorithm 1 contains a lot of contents that can be removed for more clarity. For instance line 23 describes the threshold equation which has been already discussed in Section 4.2.	**We have now restructured Algorithm 1 making it more concise and textual. We have also removed the threshold equation from the algorithm, referencing the text.**
11	A summary at the beginning of Section 5 (Experiments) will help the reader form a mindset to follow the details of the experimental content. Hence I suggest that Subsection 5.3 appears at the beginning of the section.	**We have now moved the summary at the beginning of the experimental section; it is the new Section 5.1.**
12	A full-stop "." is missing at the end of the first bullet in the Problem Statement.	**We have added the full stop.**
13	In Section 3.2, the sentence right after the contextual influence formula is repeated few times in the manuscript, hence is redundant.	**We have removed the sentence.**
14	It is recommended to add commas for numbers of more than 3 digits for more readability.	**We have added the commas.**

Sincerely,
Elisa Quintarelli, Emanuele Rabosio, and Letizia Tanca

Efficiently Using Contextual Influence to Recommend New Items to Ephemeral Groups[1] [2]

Elisa Quintarelli[3a], Emanuele Rabosio[4a,*], Letizia Tanca[a]

[a] *Politecnico di Milano – Piazza Leonardo Da Vinci 32, 20133 Milano, Italy*

Abstract

Group recommender systems suggest items to groups of users that want to utilize those items together. These systems can support several activities that can be performed together with other people and are typically social, like watching TV or going to the restaurant. In this paper we study ephemeral groups, i.e., groups constituted by users who are together for the first time, and for which therefore there is no history of past group activities.

Recent works have studied ephemeral group recommendations proposing techniques that learn complex models of users and items. These techniques, however, are not appropriate to recommend items that are new in the system, while we propose a method able to deal with new items too. Specifically, our technique determines the preference of a group for a given item by combining the individual preferences of the group members on the basis of their *contextual influence*, the contextual influence representing the ability of an individual, in a given situation, to guide the group's decision. Moreover, while many works on recommendations do not consider the problem of efficiently producing recommendation lists at runtime, in this paper we speed up the recommendation process by applying techniques conceived for the top-K query processing problem. Finally, we present extensive experiments, evaluating: (i) the accuracy of the recommendations, using a real TV dataset containing a log of viewings performed by real groups, and (ii) the efficiency of the online recommendation task, exploiting also a bigger partially synthetic dataset.

Keywords: group recommendations, influence, context, efficiency

[1] This article is an extended version of a paper presented at the 10th ACM Conference on Recommender Systems [41].

[2] Declarations of interest: none.

[3] Present address: Università di Verona, Dipartimento di Informatica (Verona, Italy)

[4] Present address: Human Technopole, Center for Analysis, Decisions and Society (Milano, Italy)

* Corresponding author.

Email addresses: elisa.quintarelli@polimi.it (Elisa Quintarelli), emanuele.rabosio@polimi.it (Emanuele Rabosio), letizia.tanca@polimi.it (Letizia Tanca)

1. Introduction

Choosing items of interest from large data catalogs that offer purchasable articles like music, video-on-demand movies, TV programs and services or products offered by online stores becomes really arduous if not supported by appropriate technologies. Indeed, a rich set of possible choices risks to confuse users and become a strain instead of an advantage.

Recommender systems [1] have been proposed to relieve the users of a tiresome task by suggesting items that might be interesting for them because they "fit their tastes". Recommender systems are employed by many important companies like Netflix, Amazon and Google [31]. In this paper we study a special class of these systems: *group recommender systems* [39], which consider the case of recommending items to be enjoyed together by a group of users.

The literature has strongly focused on recommender systems for single users; however, lots of activities are inherently social, usually carried out by multiple users: people seldom go to the movies or watch TV on their own, and equally rarely eat out alone. Even the choice of a book may be a group activity, if this book is then to be discussed and analyzed by an (online) community. Last, but not least, touristic activities are often organized for groups, thus group recommendations may provide tour organizers with useful suggestions to compose the daily schedule.

The group recommendation problem introduces further challenges with respect to the traditional single-user recommendations. First of all, often the group members have different preferences, and finding items that meet the tastes of everyone may be impossible. Moreover, people, when in a group, may exhibit different behaviors with respect to when they are alone, and therefore their individual preferences sometimes might not be a reliable source of information.

Past work on group recommender systems deals with two different kinds of groups: *persistent* and *ephemeral* [39]. Persistent groups have a significant history of activities together, while ephemeral groups are formed by people who may happen to be together for the first time[7]. While, in the case of persistent groups, classical recommendation techniques for individual users are employable because the group can be considered as a unique individual, with ephemeral groups the group preferences must be computed on the basis of those known for the members of the group. In this paper we study the (more general) scenario of ephemeral groups.

Most approaches to ephemeral group recommendations propose to obtain the preferences of the group

[7] Notice that this does not mean that the users are unknown to the system, but just that they have never been with that same group before.

by appropriately combining those of the component users (previously computed by exploiting recommendation techniques for individuals) by means of some aggregation functions. Sometimes this combination is performed using weights based on complex user and item models, built from the available feedback provided by the users themselves. However, these techniques cannot be applied when several items are new to the system, since for them the system does not possess any past feedback.

Our research refers to the *item cold start* setting, proposing an approach applicable also to new items, and therefore concentrating on aggregation functions that allow to take also these into account. This possibility is crucial in several real cases: a typical example is the TV domain, where programs are seldom shown more than once and therefore cannot be associated with past user feedback.

Also note that, since our groups are ephemeral, there is a matter of *cold start* also w.r.t. the group itself. Indeed, in our proposal we cannot take into account the past choices of the same group of people, since this is the first time they choose something together. However we rely on the knowledge of the past choices of the individual people when alone, and also when in other, different groups.

A very interesting observation is that, when people are in a group, they often make choices that they would not make when alone, since there are social factors – like the influence of other group members or the need to arrive at a mediation – that cause a modification of the choice w.r.t. the individual preferences. As a consequence, when aggregating the preferences of the members of a group, not all members should be assigned the same weight, since some might have a greater *influence* in determining the final decision [52].

Consider for instance a family with parents and children in front of the TV: clearly, the parents may be prone to favor the children's wishes in deciding the program to be watched, thus making the children more influential in the decision process. However even this is questionable, since this behavior, which is ok when choosing among children programs, is not advisable when other kinds of programs are available and the parents have to watch to exclude those inappropriate for children.

For this reason, in this work we propose a novel aggregation function, taking influence into account but also considering *influence in its context*: indeed, just like the users' preferences [2], influence weights may vary with respect to the situation, i.e., the context, that the group is currently experiencing.

Note that recommendations are normally generated at runtime, when specific (groups of) users need them. As a consequence, the design of a recommender system must seriously take into account efficiency: since the user(s) are not willing to wait minutes, or even seconds, to obtain the recommendations, the system must be responsive and guarantee a fast and fluid interaction. The efficiency of group recommendations has been studied in [4, 7, 5], which propose the application of the threshold strategy of [21] to speed up the

process. Our technique is based on the same principles, this time also encompassing the recommendation context and user influence.

In summary, the contribution of this paper is threefold:

- We propose a novel group recommendation technique where the user preferences, as well as the influence of a user on a group, are *context-aware*.
- We design an efficient algorithm making our recommendation strategy efficient in the online recommendation task.
- We demonstrate the accuracy and efficiency of the technique by means of an experimental campaign based on a large real-world dataset containing the personal and group choices of TV programs of 7,921 users for three months.

This article extends our preliminary work [41] where the intuition of the inherent context-awareness of user influence was first presented. Here, we substantially enrich the work by solving the performance-related issues and presenting further experiments.

We now introduce our running example, based on the large real dataset containing the personal and group choices of TV programs on which we performed our experiments. We consider the existence of this dataset as a veritable luck for us: it is so rich of information, and so large, that we feel quite comfortable to think that, when controlling our results against the test set, we can take the accuracy measure as a very reasonable esteem of the user satisfaction.

Running Example. In this example the context is described through a single dimension `time_slot` that may assume the values `daytime` and `night`. Moreover, there are four users, i.e., Alice (a), Bob (b), Mark (m) and Susan (s); they watch TV programs, alone or in groups.

Alice has a very strong personality, and is willing to watch her favorite programs in any case. On the contrary, Bob is particularly interested in choosing the programs during daytime, but does not like to create problems in the choice during the evening. Finally, Mark and Susan are more permissive. Table 1 shows a small sample log of choices in the two different contexts `time_slot` = `daytime` and `time_slot` = `night`. Each row indicates the time instant, the context, the composition of the group and the chosen item. Since in our scenario, in a given time instant, only a subset of the items is available (i.e., the programs that are on the air), for the readers' convenience each row reports also the items that are available in the corresponding time instant.

Table 1: A log for our running example

Time	Context (time_slot)	Group	Chosen item	Available items
t_1	daytime	{a, b, m}	p_1	$\{p_1, p_2, p_3\}$
t_2	daytime	{a, m, s}	p_5	$\{p_4, p_5, p_6\}$
t_3	daytime	{b, s}	p_8	$\{p_7, p_8, p_9\}$
t_4	daytime	{a, m}	p_{11}	$\{p_{10}, p_{11}, p_{12}\}$
t_5	night	{a, b, s}	p_{15}	$\{p_{13}, p_{14}, p_{15}\}$
t_6	night	{b, s}	p_{16}	$\{p_{16}, p_{17}, p_{18}\}$
t_7	night	{a, b}	p_{21}	$\{p_{19}, p_{20}, p_{21}\}$

Paper Structure. The paper is organized as follows. Section 2 contains the related work, while Section 3 explains our recommendation technique: offline user influence computation and online computation of top-K group recommendations. Section 4 proposes an efficient technique for online computation of the top-K group recommendations. Section 5 describes the experimental evaluation and, finally, Section 6 concludes the paper.

2. Related Work

Many authors have studied the problem of group recommendations, the Polylens system [39], by O'Connor et al. in 2001, being among the first ones. We start with reviewing *recommendations to persistent groups*, where the method can use previously gathered information (Subsection 2.1) and *recommendations to ephemeral groups*, where previous feedback is not available and the collective group interests must be induced from those of its members (Subsection 2.2). Then we describe the works considering the concept of *influence* (Subsection 2.3), and finally those dealing with efficiency issues (Subsection 2.4).

2.1. Persistent Groups

Chen et al. [13] propose a method for persistent group recommendations based on a weighted average of the group members' scores; here, the weight is based on a genetic algorithm whose fitness function relies on known group preferences.

In [48], Vildjiounaite et al. propose recommendations of TV programs to families and each family is considered as a (persistent) group. The algorithm is based on classification: one classifier is created for each family.

To find the item scores, Seko et al. [42] use the similarities with other items recorded in the group history.

The model of Hu et al. [27] is based on restricted Boltzmann machines, learned on the basis of ratings expressly provided by the groups.

The approaches above cannot be compared with ours, because in the case of ephemeral groups we cannot count on the previous group experience. Thus, let us now consider the literature on ephemeral groups.

2.2. Ephemeral Groups

Ephemeral group recommendations are obtained from the preferences of the group members, applying some aggregation function like the *average* or the *least misery*.

The works in [36, 50, 44, 6, 43, 11, 38, 12, 17, 29, 3, 24] apply the *average of the group members'* preferences for an item to compute that of the group. Other proposals rely on a variant of the plain average named *average without misery* [36, 45, 17], which is an average where the items with preference below a given threshold are discarded.

Instead, the works [39, 36, 6, 43, 11, 38, 47, 25, 12, 17, 29, 24] adopt the *least misery* technique, whose objective is to recommend the items that are "less disliked" by the group members, thus trying not to make anyone discontented. Indeed, least misery recommends the items for which the worst preference value of the group members is the highest. For instance, suppose we have a group containing two users A and B and the following preferences for the items i_1 and i_2: $p(A, i_1) = 4$, $p(B, i_1) = 6$, $p(A, i_2) = 10$, $p(B, i_2) = 1$. The least misery technique looks at the lowest scores for the items, i.e., 4 and 1, and recommends i_1 to the group.

Another common way to aggregate is the *maximum satisfaction* method [36, 11, 38, 12, 17, 29]. With this criterion, we choose the items for which the greatest value among the preferences of the group members is the highest. In the previous example, this strategy would consider the greatest scores 6 and 10, and recommend i_2 to the group.

The authors of [8, 23, 3] produce an average of the members' preferences weighted by the members' expertise, hypothesizing that, the longer the users have used the system, the more expert they are.

Very interestingly, the function proposed by Amer-Yahia et al. [4] tries to maximize the satisfaction of the group components, at the same time aiming at minimizing *disagreement*. Some kind of *fairness* among group members is also considered in [33], where the score of a top-k list for a group is obtained by integrating its relevance with respect to the preferences of the individual users with a measure of the *fairness*, to guarantee that all the users are somehow satisfied.

We employ a sophisticated kind of weighted average, more complex than the just described aggregation functions. Our experiments use the above works as the main references for comparison.

De Campos et al. [16] propose a Bayesian network to learn both the individual preferences and the aggregation weights; unfortunately this method cannot be used to recommend new items, because it requires

to build a complex model including items already known.

2.3. Influence in Group Recommendation

Masthoff and Gatt have first acknowledged the significance of influence in [37], where they envisage two kinds of influence: *emotional contagion*, related to the affective position of a user w.r.t. the other group members, and *conformity*, concerning the tendency of a member of the group to adapt to the choices of the others. These factors are computed starting from information on the relationships of people within the group, and integrated in the group preference.

Quijano-Sanchez et al. [40] combine individual preferences on the basis of *personality* and *trust* of the group members: a user with a higher personality factor has greater weight, while the group members' preferences are adapted to be similar to those of the users whom they trust. In order to derive these two components, the authors perform psychological tests and extract information from social networks.

Personality, represented by means of a weighted graph indicating how the users influence each other, is used by Kompan and Bielikova [30] too: also here the influence values are produced by means of psychological tests.

The social aspects considered by Christensen and Schiaffino [14] are *trust*, *social similarity* and *social centrality*. The first relates to the degree of familiarity between pairs of users, defined in advance, while the second and the third ones are based on information coming from social networks. Amer-Yahia et al. [5] propose to rely on *affinity*, that is, the degree of connection between two users; again, the authors derive affinities from social network information. These approaches exploit social factors, which needs a prior knowledge of their values. By contrast, our method computes influences only on the basis of training data.

Some approaches build complex models of users and items to compute both the individual preferences and the influence values needed to calculate the aggregations. In [49], a probabilistic model learned with an expectation-maximization procedure is adopted, while Liu et al. [34] and Yuan et al. [51] use latent Dirichlet allocation. Finally, Kim and El Saddik [28] build a graph for each group containing users and items, and use random walks with restarts to compute affinities between users and items, and the influence of the users within the groups.

These methods can only be used with items that are already known to the system, since previously unknown items cannot be included in the learned model.

Summarizing, our research proposes an aggregation function that derives group preferences from individual ones, introducing the users' influence factor so that it works also for new items. Moreover, our influence values are determined using only the training data, so that no interaction with the users is needed. Finally,

to the best of our knowledge, ours is the first proposal taking into account the fact that the *current context* can modify the influence exerted by the single group members.

2.4. Efficiency of Group Recommenders

The problem of efficiency, certainly not secondary in recommender systems, has first been tackled for the case of group recommendations in [4], which improves the performance of the disagreement-based strategy on the basis of the Fagin's Threshold Algorithm [21]. The idea of this paper is to rely on the fact that, since the procedure should only produce a small number of items (top-K), it is not necessary to look at all the available ones. However, the algorithm proposed in [4] is only optimized w.r.t. time, so the same authors extend their work in [7], focusing on the optimization of space. Fagin-style algorithms are proposed also for the efficient implementation of the affinity-based methodology of [5].

We also adopt the Fagin's thresholding mechanism in our group recommendation strategy and, similarly to [7], we analyze how to reduce the amount of storage devoted to precomputed information, however without significantly affecting the execution time.

3. Group Recommendation Technique

This section describes our group recommendation strategy. Our approach takes as input the individual preferences of the users for the items, and provides techniques to aggregate them in order to obtain group preferences. We set ourselves in the general framework of *context-aware* recommender systems and suppose that the preferences of users for items may vary according to their current contexts; therefore each user preference is associated with the context in which it holds. We also suppose we have a log recording the history of the items chosen in the past by groups of users and, to be more general, assume that the set of the items available for the users' choices may change with time. This is the case of the TV setting of the running example, but also of other scenarios like event recommendations or recommending movies shown in theaters now.

The description of our strategy proceeds as follows. Subsection 3.1 introduces the context model we employ, while Subsection 3.2 presents the data model and the problem statement. The details of our recommendation technique are given in Subsection 3.3.

3.1. Context Model

The *context* is constituted by any information that can be used to characterize the situation of an entity [18]. In particular, in the recommender scenario it represents the situation the users are going through when choosing the items [2].

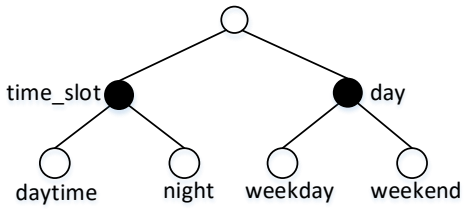

Figure 1: Tree-based representation of the `time_slot` and `day` context dimensions

The available contexts in a given scenario are described here through a set of n dimensions, each in its turn associated with a set of possible values [10]. Given the available dimensions and their values, a context is defined as a set of dimension/value pairs:

$$c = \texttt{dimension}_1 = \texttt{value}_1 \wedge \ldots \wedge \texttt{dimension}_n = \texttt{value}_n \tag{1}$$

In our running example there is just one dimension, i.e., the `time_slot`, with two possible values: `daytime` and `night`. Another dimension usually applicable within recommendation systems is the `day` of the week; possible values for such a dimension are `weekday` and `weekend`. With these two dimensions, the possible contexts are obtained by combining the values of `time_slot` and `day`; for instance, a valid context is $\texttt{time\_slot} = \texttt{daytime} \wedge \texttt{day} = \texttt{weekend}$. Figure 1 provides an intuitive tree-based representation of the `time_slot` and `day` context dimensions and their values.

In several applications the context is considered as hierarchical [10]. With our model this can be obtained in two ways: (i) some values could admit subdimensions, e.g., the `night` value in our example could admit a subdimension named `night_part`, with values `early_night` and `late_night`. So, $\texttt{time\_slot} = \texttt{night} \wedge \texttt{day} = \texttt{weekend}$ would be more general than $\texttt{night\_part} = \texttt{early\_night} \wedge \texttt{day} = \texttt{weekend}$; (ii) not all dimensions must necessarily have a value, for example the context $\texttt{time\_slot} = \texttt{daytime}$ is actually more general than $\texttt{time\_slot} = \texttt{daytime} \wedge \texttt{day} = \texttt{weekend}$.

We use the symbol $\succ$ to denote this hierarchical relationship, and the relation $\succeq$ is also naturally defined as follows: for every $c_i, c_j \in \mathcal{C}$, $c_j \succeq c_i$ if and only if $c_j \succ c_i$ or $c_j = c_i$.

Note that here, in order to concentrate on the original aspects of this work (ephemeral group recommendation techniques) in our discussions and examples we consider simple contexts, where all the dimensions are instantiated. The only hierarchical aspect is given by the presence of a *universal context* $\tilde{c}$, which is a special context that applies in all the situations determined by all the other contexts, so it is *more general* than all the other contexts. $\tilde{c}$ can be seen as associated with the root node of the tree in Figure 1 and, for

every context c_i, $\tilde{c} \succ c_i$.

3.2. Data Model and Problem Statement

We consider a set of items $\mathcal{I}$ and a set of users $\mathcal{U}$; any group $G \in \wp(\mathcal{U})$ can be extracted from the set of users. $\mathcal{C}$ is the set of possible contexts in the given scenario, where a context in $\mathcal{C}$ is either a set of dimension/value pairs, as in Equation (1), or the universal context $\tilde{c}$. Every item access happens at a time instant t, in which just a subset of the items in $\mathcal{I}$ is available.

$\mathcal{L}$ is the log recording the history of the items previously chosen by groups formed in the past, used to compute the influence of the various users; each element of the log is a 4-ple (t_j, c_j, G_j, i_j) where t_j is the time instant in which the item has been chosen, $c_j \in \mathcal{C}$, $G_j \in \wp(\mathcal{U})$ and $i_j \in \mathcal{I}$.

$\mathcal{H}$ is the log of the past individual choices; we do not specify the details of its elements because they depend on the specific method chosen for the computation of individual preferences[8], a topic out of the scope of this paper.

$score(u, i, c)$, with $u \in \mathcal{U}$, $i \in \mathcal{I}$, $c \in \mathcal{C}$, is the context-aware scoring function, assigning, for each user, a score given to the items in the various contexts. In the following it will become clear that also *context-unaware* preferences will turn out to be useful; for notational convenience, the non-contextual preferences are associated with the universal context $\tilde{\mathtt{c}}$.

$TopK(u, c, t)$ is the list of the K items preferred by user u in context c, according to the values of $score(u, i, c)$ for the each $i \in \mathcal{I}$ available at instant t. For instance, in the TV domain of our running example, the $TopK$ at a certain time instant may include only programs on the air at that time instant.

In the following we will formally define the *contextual influence* of user u in context c, indicated by $infl(u, c)$. The notation $infl(u, \tilde{c})$ is used to denote the non-contextual influence of the user u.

Problem statement. *Let $\mathcal{U}$ be a set of users, $\mathcal{I}$ a set of items, $\mathcal{C}$ a set of contexts including also the universal context $\tilde{\mathtt{c}}$. Supposing that the following data are known:*

- *The context-aware scoring function $score(u, i, c)$, $u \in \mathcal{U}$, $i \in \mathcal{I}$, $c \in \mathcal{C}$.*
- *The log $\mathcal{L}$ recording the history of the items chosen by all past groups.*

Then, given a target group $G \in \wp(\mathcal{U})$, a context $c \in \mathcal{C}$ and a time instant t, the problem of group recommendation is defined as recommending to the users in G a list (i.e., an ordered set) of K items, considered interesting in context c, from those items in $\mathcal{I}$ that are available at time instant t.

[8] We give a quick account of individual preference computation in Section 3.3.1.

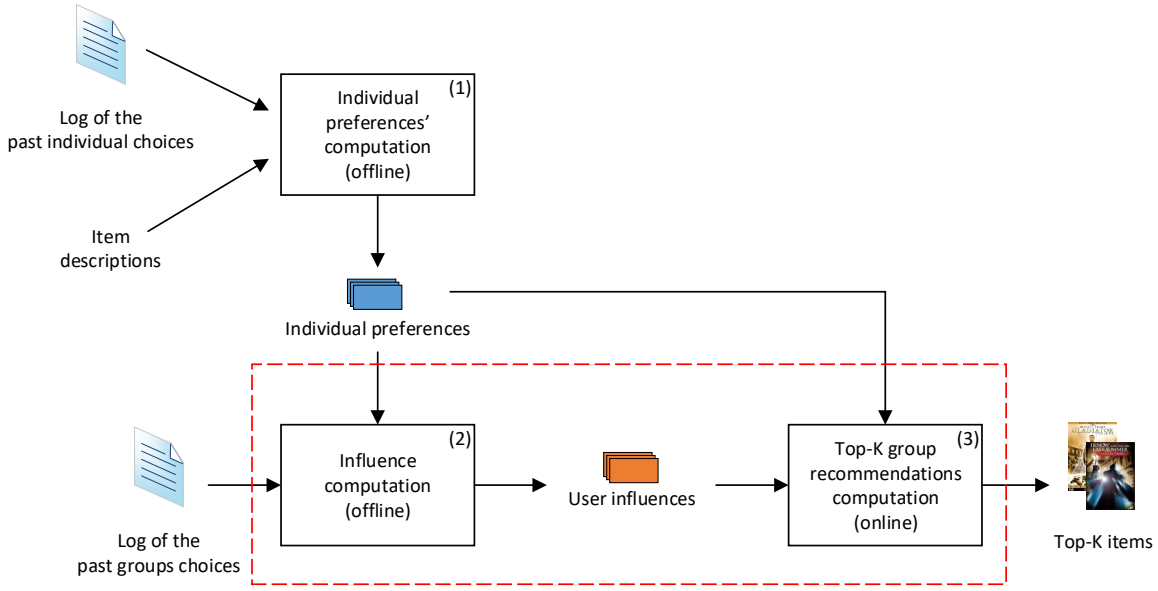

Figure 2: Phases of our recommendation strategy

3.3. Computation of the Recommendations

The phases leading to the computation of the top-K recommendation lists are summarized in Figure 2. Phases 1 and 2 are performed offline, and determine the individual preferences and the user influences. Phase 3 deals with the computation of the group recommendations, and is performed online, when required by some group of users. Note that this paper focuses on Phases 2 and 3 (in the dashed rectangle) while the individual preferences derived in Phase 1 – as specified in the problem statement above – are supposed to be provided as input. The following subsections describe each phase in detail.

3.3.1. Individual Preference Computation

Even if the details of this phase are out of the scope of the paper, to better explain our technique this subsection gives a short account of it. The individual preferences are computed offline on the basis of the log of the past individual choices and on the basis of the item descriptions in terms of their attributes (e.g., channel and genre of a TV program), using any recommender system for single users from the literature. The contextual preferences can be computed using a context-aware recommender system, while the non-contextual ones – to be associated with the universal context $\tilde{c}$ – can be derived through a traditional, context-unaware recommender system. Since we aim to deal also with new items, we suppose that the individual preferences for the items be determined through a content-based recommender, which derives them by combining the preferences for the item attributes. In the case of space efficiency problems, we can also suppose that, instead of storing the preferences of each user for all the items, only the preferences *for*

the values of the item attributes, or for their combinations, be stored. So, for instance, given a user, instead of storing their preferences for all the available TV programs, we store just their preferences for the channels and genres, whose number is obviously much smaller. Then, when needed, the preferences for the items can be easily computed on the fly exploiting the preferences for the item attributes.

Let us now review two possible content-based techniques suitable for the computation of individual preferences, both contextual and non-contextual.

A common content-based method to compute individual preferences relies on the Naive Bayes classifier (surveyed for instance in [35]). The score of every attribute value for each user (in each context, if needed) is computed as the probability of occurrence of that value in the items accessed by that user (in that context); the score of an item is then determined by aggregating the probabilities of the attribute values associated with that item using the Bayes formula. Let a be the number of attributes and v the number of possible values for each attribute. The offline computation of the scores of the attribute values can be performed scanning the log once, in $O(|\mathcal{H}|)$ time, while the online computation of the item scores through the Bayes formula requires $O(a)$ time. To store the preferences for the attribute values for each user and context $O(a \cdot v|\mathcal{U}||\mathcal{C}|)$ space is necessary.

A state-of-the-art content-based technique conceived for the TV domain is that proposed in [15], which will be employed in the experimental evaluation of Section 5. This technique builds a multidimensional tensor where the dimensions are represented by users, context dimensions and item attributes. Each element of the tensor, identified by a specific user and by an assignment of values to the context dimensions and item attributes, is filled with the number of seconds of viewing of items with those attribute values performed by the user in that context. The score of an item for a user in a context is then obtained by simply retrieving the tensor element corresponding to the appropriate user, context and attribute values. The offline computation of the tensor can be again performed in $O(|\mathcal{H}|)$, while online we need just to retrieve the appropriate tensor element, which can be done in constant time. The tensor maintains a preference for each combination of the attribute values, so the space needed to store the tensor is $O(v^a|\mathcal{U}||\mathcal{C}|)$; here the required space is exponential in the number of attributes, so this strategy is not advisable when the number of attributes is not negligible.

3.3.2. Influence Computation

The group preference for an item is obtained by aggregating the individual preferences of the group members *on the basis of their influence*. In each context c, the influence $infl(u, c)$ of a given user u is derived offline by comparing the behavior of u when alone (i.e., u's individual preferences) with u's behaviors in

groups (i.e., the interactions contained in the log $\mathcal{L}$). Basically the influence of u tells *how many times the groups containing u have selected one of u's favorite items.*

Let $TopK(u, c, t)$ be the list of the K items preferred by user u in context c, according to the values of $score(u, i, c)$ for the each $i \in \mathcal{I}$ available at instant t[9]. The contextual influence is defined as follows:

$$infl(u, c) = \frac{|l_j \in \mathcal{L} : c \succeq c_j \wedge u \in G_j \wedge i_j \in TopK(u, c, t_j)|}{|l_j \in \mathcal{L} : c \succeq c_j \wedge u \in G_j|} \tag{2}$$

The computation of the influence values requires to scan the log of the past groups activities once, so it can be performed in $O(|\mathcal{L}|)$ time. Moreover, we need to store an influence value for each user and context, therefore the space needed is $O(|\mathcal{U}||\mathcal{C}|)$.

The value of $infl(u, c)$ quantifies the ability of user u to direct the group's decision towards u's own tastes while in context c. Note that the non-contextual influences, associated with the universal context $\tilde{c}$, are computed considering the whole log.

Example 3.1 (Influence)**.** *Suppose we compute the influences using $K = 1$ in Equation (2), i.e., using top lists of just one item, and suppose that at the time instants of Table 1 the programs preferred by the users be the following:*

- t_1*:* $a \rightarrow p_2$, $b \rightarrow p_1$, $m \rightarrow p_2$
- t_2*:* $a \rightarrow p_5$, $m \rightarrow p_5$, $s \rightarrow p_4$
- t_3*:* $b \rightarrow p_8$, $s \rightarrow p_9$
- t_4*:* $a \rightarrow p_{11}$, $m \rightarrow p_{10}$
- t_5*:* $a \rightarrow p_{15}$, $b \rightarrow p_{13}$
- t_6*:* $b \rightarrow p_{16}$, $s \rightarrow p_{16}$
- t_7*:* $a \rightarrow p_{21}$, $b \rightarrow p_{20}$

that is, at time t_1 Alice prefers p_2, Bob prefers p_1 and Mark prefers p_2, and so on. The contextual influence computations for users Alice, Bob and Mark are as follows:

$infl(a, \texttt{time\_slot} = \texttt{daytime}) = 2/3 = 0.667$

$infl(b, \texttt{time\_slot} = \texttt{daytime}) = 2/2 = 1$

$infl(m, \texttt{time\_slot} = \texttt{daytime}) = 1/3 = 0.333$

[9]For instance, in the TV domain of our running example, the *TopK* at a certain time instant may include only programs on the air at that time instant.

$infl(a, \texttt{time\_slot} = \texttt{night}) = 2/2 = 1$

$infl(b, \texttt{time\_slot} = \texttt{night}) = 1/3 = 0.333$

$infl(a, \tilde{\texttt{c}}) = 4/5 = 0.8$

$infl(b, \tilde{\texttt{c}}) = 3/5 = 0.6$

$infl(m, \tilde{\texttt{c}}) = 1/3 = 0.333$

The non-contextual influences, associated with the universal context $\tilde{\texttt{c}}$*, are computed using the whole log. Moreover, the value of* $infl(m, \texttt{time\_slot} = \texttt{night})$ *cannot be computed, because Mark provided no feedback in the context* $\texttt{time\_slot} = \texttt{night}$*.*

3.3.3. Top-K Group Recommendation Computation

Top-K recommendations are computed online, when a group of users requires that the system suggests some interesting items to be enjoyed together. The system must compute the group preferences for the items, and then determine the K items with the highest scores.

Given a group $G \in \wp(\mathcal{U})$, its preference $score(G, i, c)$ for $i \in \mathcal{I}$ in the context $c \in \mathcal{C}$ is computed as the average of the preferences of its members weighed on the basis of each member's influence in context c:

$$score(G, i, c) = \frac{\sum_{u \in G} infl(u, c) \cdot score(u, i, c)}{\sum_{u \in G} infl(u, c)} \tag{3}$$

Please notice that, if the number of contexts is high, it may well happen that some users are not associated with feedbacks in some contexts. In these cases, the non-contextual preferences (associated with the universal context $\tilde{\texttt{c}}$) can be used.

Then, the top-K list of items preferred by a certain group G in context c at time instant t is determined as follows:

1)
- If all the members of G have provided feedback in context c, compute a context-aware score for all the items using Equation (3).
- Otherwise, compute a score for all the items using, in Equation (3), the universal context $\tilde{\texttt{c}}$ instead of the context c.

2) Retrieve the K items with the greatest scores among those available at time t.

We call this recommendation method *CtxInfl*.

Note that, initially, to determine the group preferences for contexts when some of the group members have not provided feedback, we thought of using non-contextual values of preference and influence *just for*

those users for whom the contextual values are not known, thus mixing context-aware and context-unaware quantities in the computation. In this way, Step 1 above would be replaced by the following:

1') Compute a context-aware score for all the items using Equation (3). If a user $u' \in G$ has not provided feedback in context c, replace in Equation (3) *score(u', i, c)* and *infl(u', i, c)* with *score(u', i,* $\tilde{\texttt{c}}$*)* and *infl(u', i,* $\tilde{\texttt{c}}$*)*.

We dub this alternative strategy *CtxInflMix*. However we will see in Section 5 that CtxInflMix shows a really bad behavior in the experiments, so we propose the alternative that does not mix contextual and non-contextual quantities. In the following example we show the behavior of both CtxInfl and CtxInflMix.

Example 3.2 (Score computation)**.** *Suppose that at instant* t_8*, during daytime, a group composed of Bob and Mark is watching TV, and they want a recommendation for a program among the available ones:* p_{22}*,* p_{23}*,* p_{24}*,* p_{25}*, and* p_{26}*. Note that this group is never recorded in the log: it is the first time that these two users watch TV together without other people. The following individual preference values are given: score(b,* p_{22}*,* $\texttt{time\_slot} = \texttt{daytime}$*) = 0.8, score(b,* p_{23}*,* $\texttt{time\_slot} = \texttt{daytime}$*) = 0.3, score(b,* p_{24}*,* $\texttt{time\_slot} = \texttt{daytime}$*) = 0.4, score(b,* p_{25}*,* $\texttt{time\_slot} = \texttt{daytime}$*) = 0.2, score(b,* p_{26}*,* $\texttt{time\_slot} = \texttt{daytime}$*) = 0.1, score(m,* p_{22}*,* $\texttt{time\_slot} = \texttt{daytime}$*) = 0.2, score(m,* p_{23}*,* $\texttt{time\_slot} = \texttt{daytime}$*) = 0.6, score(m,* p_{24}*,* $\texttt{time\_slot} = \texttt{daytime}$*) = 0.9, score(m,* p_{25}*,* $\texttt{time\_slot} = \texttt{daytime}$*) = 0.1, score(m,* p_{26}*,* $\texttt{time\_slot} = \texttt{daytime}$*) = 0.5. The group scores for the five available items are computed as weighted averages. Since both users have provided feedback in the context* $\texttt{time\_slot} = \texttt{daytime}$*, CtxInfl employs contextual preferences and influences in Equation* (3)*. Using the influence values computed in Example 3.1:*

$$score(\{b, m\}, p_{22}, \texttt{time\_slot} = \texttt{daytime}) = \frac{1 \cdot 0.8 + 0.333 \cdot 0.2}{1 + 0.333} = 0.650$$

$$score(\{b, m\}, p_{23}, \texttt{time\_slot} = \texttt{daytime}) = \frac{1 \cdot 0.3 + 0.333 \cdot 0.6}{1 + 0.333} = 0.375$$

$$score(\{b, m\}, p_{24}, \texttt{time\_slot} = \texttt{daytime}) = \frac{1 \cdot 0.4 + 0.333 \cdot 0.9}{1 + 0.333} = 0.525$$

$$score(\{b, m\}, p_{25}, \texttt{time\_slot} = \texttt{daytime}) = \frac{1 \cdot 0.2 + 0.333 \cdot 0.1}{1 + 0.333} = 0.175$$

$$score(\{b, m\}, p_{26}, \texttt{time\_slot} = \texttt{daytime}) = \frac{1 \cdot 0.1 + 0.333 \cdot 0.5}{1 + 0.333} = 0.200$$

Therefore, the system recommends p_{22} *to the group. Note that the choice is guided by the heavier influence of Bob.*

Suppose now that at instant t_9*, in the context* $\texttt{time\_slot} = \texttt{night}$*, a group with Alice, Bob and Mark is watching TV and needs a recommendation, choosing among* p_{27}*,* p_{28} *and* p_{29}*. Since Mark provided no feedback in the context* $\texttt{time\_slot} = \texttt{night}$*, CtxInfl uses non-contextual preferences and influences. The following preferences are given: score(a,* p_{27}*,* $\tilde{\texttt{c}}$*)=0.9, score(a,* p_{28}*,* $\tilde{\texttt{c}}$*)=0.1, score(a,* p_{29}*,* $\tilde{\texttt{c}}$*)=0.6, score(b,*

p_{27}, $\tilde{c}$)=0.2, score(b, p_{28}, $\tilde{c}$)= 0.9, score(b, p_{29}, $\tilde{c}$)=0.3, score(m, p_{27}, $\tilde{c}$)=0.3, score(m, p_{28}, $\tilde{c}$)= 0.7, score(m, p_{29}, $\tilde{c}$)=0.5. The group scores for the three available items are computed using the non-contextual influences of Example 3.1 and the non-contextual preferences:

$$score(\{a,b,m\}, p_{27}, \texttt{time\_slot} = \texttt{night}) = \frac{0.8 \cdot 0.9 + 0.6 \cdot 0.2 + 0.333 \cdot 0.3}{0.8 + 0.6 + 0.333} = 0.542$$

$$score(\{a,b,m\}, p_{28}, \texttt{time\_slot} = \texttt{night}) = \frac{0.8 \cdot 0.1 + 0.6 \cdot 0.9 + 0.333 \cdot 0.7}{0.8 + 0.6 + 0.333} = 0.492$$

$$score(\{a,b,m\}, p_{29}, \texttt{time\_slot} = \texttt{night}) = \frac{0.8 \cdot 0.6 + 0.6 \cdot 0.3 + 0.333 \cdot 0.5}{0.8 + 0.6 + 0.333} = 0.477$$

Therefore, the system recommends program p_{27} to the group.

Suppose now we employ CtxInflMix instead of CtxInfl with the group constituted by Bob and Mark during daytime. CtxInflMix behaves exactly like CtxInfl at instant t_8, because contextual feedback is available from both users. At time instant t_9, on the contrary, a recommendation is required for Alice, Bob and Mark during the night, and Mark's feedback during the night is missing. In this case CtxInflMix uses contextual preferences and influences for Alice and Bob, and non-contextual preferences and influence – referred to the universal context $\tilde{c}$ – for Mark. Suppose that the following preferences be given during the night: score(a, p_{27}, $\texttt{time\_slot} = \texttt{night}$)=0.4, score(a, p_{28}, $\texttt{time\_slot} = \texttt{night}$)=0.3, score(a, p_{29}, $\texttt{time\_slot} = \texttt{night}$)=0.5, score(b, p_{27}, $\texttt{time\_slot} = \texttt{night}$)=0.4, score(b, p_{28}, $\texttt{time\_slot} = \texttt{night}$)= 0.5, score(b, p_{29}, $\texttt{time\_slot} = \texttt{night}$)=0.6. The group scores for the three available items are computed as follows:

$$score(\{a,b,m\}, p_{27}, \texttt{time\_slot} = \texttt{night}) = \frac{1 \cdot 0.4 + 0.333 \cdot 0.4 + 0.333 \cdot 0.3}{1 + 0.333 + 0.333} = 0.380$$

$$score(\{a,b,m\}, p_{28}, \texttt{time\_slot} = \texttt{night}) = \frac{1 \cdot 0.3 + 0.333 \cdot 0.5 + 0.333 \cdot 0.7}{1 + 0.333 + 0.333} = 0.420$$

$$score(\{a,b,m\}, p_{29}, \texttt{time\_slot} = \texttt{night}) = \frac{1 \cdot 0.5 + 0.333 \cdot 0.6 + 0.333 \cdot 0.5}{1 + 0.333 + 0.333} = 0.520$$

Therefore, CtxInflMix recommends program p_{29} to the group.

4. Efficient Online Computation of the Group Recommendations

Once an accurate group recommendation strategy has been designed, the subsequent step is to make it efficient, to be able to recommend top-K lists on the fly.

The on-the-fly computation of the top-K recommendation lists is executed by Phase 3 in Figure 2, taking as input individual preferences and influences. In Section 3.3.3 the computation of the top-K recommendations for a group is performed by determining the group preferences for all the items, and then by choosing the K items with the highest scores. In this section, we concentrate on this phase, and propose an algorithm to make the online computation of the top-K recommendations faster.

In particular, since the number of items may be very large and significantly bigger than K, computing all

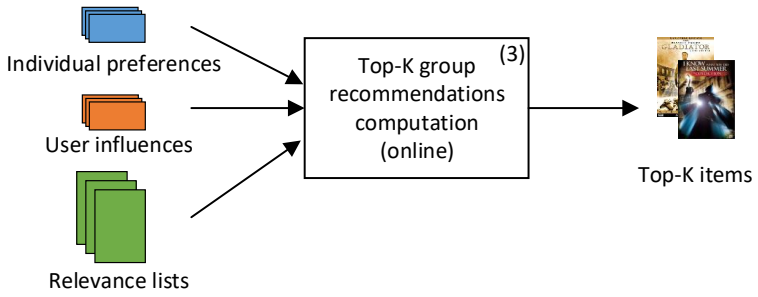

Figure 3: Phase 3 revised

the ranks may be highly inefficient. In such situations the top-K list may be computed more efficiently by conveniently extending the information that is precomputed offline and given in input to Phase 3 with the *relevance lists* associated with each pair (user, context) (see Figure 3). The relevance list associated with user u and context c is the list of the items in decreasing order with respect to the individual contextual preferences of u in context c. Once the relevance lists are available, we exploit the principles at the basis of the Threshold Algorithm (TA) of [21] to make the computations faster.

In the rest of this section we start by overviewing the TA thresholding principle (Subsection 4.1), and then explain how this mechanism can be applied to our scenario (Subsection 4.2).

4.1. TA Thresholding Principle

The TA algorithm, proposed by Fagin et al. [21], introduces a thresholding principle to make top-K computations faster. The aim of TA is to quickly determine the top-K items in the situations where the score of an item is derived by aggregating the scores of individual components and the relevance lists associated with the individual components are available. The relevance list of a component is the list of items in decreasing order on the basis of their scores for that component.

TA can be applied to several domains. For instance, the items we want to rank may be tuples of a relational database, with the individual components concurring to the score being the tuple attribute values, whose scores will be aggregated to compute the score of a tuple. Another example are text documents, where the score of a document can be computed by aggregating the scores of the keywords included in the document; in this case, the keywords play the role of the individual components.

For example, consider the document ordering case. Suppose that there are four documents d_1, d_2, d_3 and d_4, and two keywords w_1 and w_2. Each keyword is associated with a relevance list, containing the documents in decreasing order on the basis of their scores for that keyword. Sample relevance lists, in which besides the ordered documents we include also their scores, are the following: w_1: {(d_3, 0.9), (d_1, 0.6), (d_2,

0.2), (d_4, 0.1)}; w_2: {(d_2, 0.8), (d_3, 0.7), (d_4, 0.3), (d_1, 0.1)}.

TA proceeds by scanning the relevance lists of the individual components in parallel, computing the aggregated score for all the items that are met. To avoid processing unnecessary items, TA applies a thresholding principle. Let *aggr* be the aggregation function employed to aggregate the scores of the individual components of an item. After the items at position j in the relevance lists have been processed, a threshold is computed aggregating through *aggr* the component scores of the items at position j in the relevance lists; note that to compute the threshold we are aggregating component scores referred to different items. If the top-K list contains K items and the lowest score currently in the list exceeds the threshold, then the authors prove that it is useless to analyze the remaining items. This thresholding principle is applicable when the aggregation function *aggr* is monotone, i.e., if $aggr(x_1, \ldots, x_m) \leq aggr(y_1, \ldots, y_m)$ whenever $x_h \leq y_h$ for every $h \in \{1, \ldots, m\}$.

Let us go on with the document ranking example, supposing that K=1 and the *aggr* function is the arithmetic average, which is monotone. TA starts scanning the documents at position 1 in the relevance lists and computes their aggregated scores: 0.8 for d_3 and 0.5 for d_2. Then, the threshold is computed as 0.85, i.e., the average of 0.9 and 0.8. Since the current top item has a score lower than the threshold, TA proceeds by considering the documents at position 2 in the relevance lists, computing their aggregated scores: 0.35 for d_1, while d_3 is ignored because it has already been analyzed. The new threshold is 0.65; since now the score of the top item is greater than the threshold, the algorithm terminates by outputting document d_3 without considering the remaining items.

4.2. Efficient Recommendation Algorithm

Let us now see how the TA principles can be applied to our scenario. Here, the user preferences are context-aware, so we have to compute a relevance list for each pair (user, context); these lists must be built offline and added to the input of the online top-K recommendation algorithm. In fact, for each context, the users can be considered as the individual components whose preferences have to be aggregated. Making a parallel with the document ranking example above, the items play the role of the documents, and the users that of the keywords. Moreover, our aggregation function – a weighted average – is clearly monotone.

As in the original TA, to compute the top-K for a certain group in a given context we maintain a list of the top-K items found so far and scan in parallel the relevance lists of each group member in each given context. When an item with a score that is greater than the lowest one in the top-K is met, it is added to the top-K list replacing the worst one. Once all the items at position j in the relevance lists have been processed, the TA thresholding principle is applied.

In more detail, let RL_{uc} be the relevance list of user u in context c and $RL_{uc}(j)$ the j-th element in this list. The threshold after all the items at position j in the relevance lists have been processed is as follows:

$$threshold_j = \frac{\sum_{u \in G} score(u, RL_{uc}(j), c) \cdot infl(u, c)}{\sum_{u \in G} infl(u, c)} \tag{4}$$

According to the TA thresholding principle, if the top-K list contains exactly K items and all of them have a score higher than the threshold, then no further items are analyzed.

To speed up the process when K is large, similarly to [4], we employ a min-heap instead of a list to maintain the top-K items. A min-heap is a tree-based data structure conceived to store ordered sets of objects. Indeed, the min-heap allows constant time access to the root element (which is that with the lowest score) and requires logarithmic time to add and remove elements. The min-heap guarantees that the root element be always the one with the lowest score, but the remaining elements are not ordered, which is sufficient in our case.

Moreover, note that the described technique requires that the relevance lists for each pair (user, context) be added to the inputs of the online top-K recommendation algorithm, and therefore precomputed offline and stored. When it is needed to save disk space, we allow to store just the upper part of the relevance lists, e.g., the top 30% items.

Algorithm 1 provides an efficient computation of the top-K group recommendations relying on contextual user influence. It takes as input the group needing the recommendations, the current context, the relevance lists of the group members, their individual preferences and influence values. As explained above, the relevance lists may have a length lower than the total number of items, in order to save disk space; we suppose that all the relevance lists have the same length. Moreover, if at the time instant t when the recommendations are required only a subset of the items is available, just those items must be included in the relevance lists.

The algorithm starts by determining whether, according to what is explained in Section 3.3.3, it is possible to use the contextual preferences and influences (Lines 2-4). This task is performed with the help of the ChooseContext function, given in Algorithm 2; Algorithm 2 provides two versions of ChooseContext, the first realizes the CtxInfl variant and the second realizes the CtxInflMix variant of our method (see Section 3.3.3). Then, Algorithm 1 proceeds initializing the min-heap (Line 5); our min-heap contains pairs <item, score>, where the score is used for the ordering. The *while* loop at Line 7 scans the relevance lists in parallel until there are no more available items, or the heap is full and the threshold is reached. When

an item that has not been processed yet is met, if the min-heap contains less than K items, then the item is added to the min-heap (Line 13); if the min-heap is full but the group score (computed through Equation (3)) is greater than the lowest one in the min-heap, then the item is inserted replacing the item with the lowest score currently in the min-heap (Line 15). After all the items at a given position in the relevance lists have been processed, at Line 19 the threshold is computed using Equation (4). If the *while* loop ends because the threshold has been reached, then the algorithm terminates returning the top-K list; on the contrary, if the loop ends because all the items stored in the relevance lists have been processed but the threshold has not been reached, there is no guarantee that the min-heap really contains the best K items, so all the remaining ones, which have not been included in the relevance lists to save disk space, must be processed (Lines 21-23). Note that the min-heap contains the best K items, but they are not sorted. Thus, if an ordered list of K items is required a sorting algorithm must be applied (Line 24).

Algorithm 1 Efficient online computation of group recommendations relying on contextual user influence

```
1: function ComputeTopK(G ∈ ℘(U), c ∈ C, relevance list RL_uc for each u ∈ G, relevance list RL_uc̃ for each
   u ∈ G, individual preferences, influence values)
2:     for all u ∈ G do
3:         c_u = ChooseContext(G, u, c)
4:     end for
5:     topkMinHeap is the min-heap storing the top-K
6:     processedItemsSet is a set containing the items processed so far
7:     while relevance lists not ended ∧ (topkMinHeap.size< K ∨ score of root of topkMinHeap ≤ threshold) do
8:         for all u ∈ G do
9:             item = next item from RL_{uc_u}
10:            if item∉ processedItemsSet then
11:                Add item to processedItemsSet
12:                if topkMinHeap.size<K then
13:                    Insert <item, score(G, item, c_u)> into topkMinHeap
14:                else if score(G, item, c_u) > score of the root of topkMinHeap then
15:                    Replace the root of topkMinHeap with <item, score(G, item, c_u)>
16:                end if
17:            end if
18:        end for
19:        Compute threshold with Equation (4)
20:    end while
21:    if threshold was not reached then
22:        Process all the remaining items
23:    end if
24:    topkList = compute (ordered) top-K list from topkMinHeap // Order only if ordered top-K list required
25:    return topkList
26: end function
```

Let us now evaluate the time complexity of the algorithm in the worst case. Let n be the number of items ($n < |\mathcal{I}|$ when not all the items are available at the current time instant) and a the number of item attributes (e.g., genre and channel of a TV program). We consider that the computation of one item score starting

Algorithm 2 Choice of the context to be used for a user with CtxInfl and CtxInflMix

function ChooseContext($G \in \wp(\mathcal{U})$, $u \in G$, $c \in \mathcal{C}$) // *CtxInfl*
    if all users in G provided feedback in context c **then**
        return c
    else
        return $\tilde{c}$
    end if
end function

function ChooseContext($G \in \wp(\mathcal{U})$, $u \in G$, $c \in \mathcal{C}$) // *CtxInflMix*
    if u provided feedback in context c **then**
        return c
    else
        return $\tilde{c}$
    end if
end function

from the scores of its attributes can be performed in $O(a)$ time. The algorithm analyzes at most n items, and for each item it may need to (i) check its membership in the set of the processed items (constant time), (ii) compute the group score, by computing the individual scores ($O(|G|a)$) and combining them ($O(|G|)$), (iii) add the item to the min-heap or replace the root ($O(\log K)$). Also the threshold computation is executed at most n times, and requires $O(|G|a)$ time. Finally, if an ordered list is required, the items in the min-heap can be ordered in $O(K \log K)$. Globally, considering a as negligible, the worst case complexity of the algorithm is $O(n|G| + n \log K + K \log K)$. Note that, using a standard list instead of a heap, the complexity would have been $O(n|G| + nK)$, so the heap is helpful when K is not negligible. The thresholding principle applied by Algorithm 1 allows to analyze $n' < n$ items, thus reducing the time to $O(n'|G| + n' \log K + K \log K)$. We will show in the experimental section (Section 5.3) that when there are many items the gain may be very relevant.

The group size $|G|$ is often negligible with respect to the number of items. For instance, in the real TV dataset employed in the experiments of Section 5 the average group size is 2.26. Also the other domains studied in the literature are associated with rather small average group sizes, even if with some differences: for place visit the reported values are 3.98 [34] and 4.68 [51], for online events 97.64 [34], for real events 10.75 [34] and 20.3 [51], and for online communities of interest 31 [28]. However, it seems reasonable to assume that in certain domains, like online events and communities of interest, the size of the group may grow consistently. In such cases, the usage of an efficient algorithm like ours would become even more crucial.

Let us now consider the space efficiency. Let v be the number of values for an item attribute. To run Algorithm 1, the following data need to be stored: (i) the relevance lists ($O(|\mathcal{I}||\mathcal{U}||\mathcal{C}|)$); (ii) the preferences for the item attribute values ($O(a \cdot v|\mathcal{U}||\mathcal{C}|)$); (iii) the influence values ($O(|\mathcal{U}||\mathcal{C}|)$). Considering a and v as

User *Bob*	
Item	**Score**
p_{22}	0.8
p_{24}	0.4
p_{23}	0.3

User *Mark*	
Item	**Score**
p_{24}	0.9
p_{23}	0.6
p_{26}	0.5

Figure 4: Relevance lists for the users Bob and Mark in context `time_slot` = `daytime`, used in Example 4.1

negligible, the required space is $O(|\mathcal{I}||\mathcal{U}||\mathcal{C}|)$. As mentioned above, we can limit the space by storing just a portion of the relevance lists, thus reducing the $|\mathcal{I}|$ term. Please notice that in some circumstances the storage space may actually be an issue. Consider for instance a dataset with size similar to the well-known Last.fm [9], with 1 million items and 300,000 users. Suppose that the contextual dimensions are the time slot and the day of the week, with 8 time slots and 7 days and therefore a total of 56 different contexts. If an entry of the relevance list requires 8 bytes to be stored, the total disk space needed is about 120 TB. Such a size might be troublesome, also considering that users, items and contexts might be even more. In such a situation, mitigating the problem by storing just the upper portion of the relevance lists may be an important help. We remark that, as explained in Section 3.3.1, it is not required that the preferences for all the items be stored, because they can be easily derived on the basis of the preferences for the item attributes.

Example 4.1 (Efficient computation of the group recommendations)**.** *Let us now efficiently compute the contextual group recommendations of Example 3.2 using Algorithm 1. The algorithm receives as input the relevance lists given in Figure 4, for the users Bob and Mark, related to the context* `time_slot` = `daytime`*. For the sake of the example, we include in the relevance lists also the scores of the items, which actually could be computed on the fly from the scores of the item attributes (e.g., genre and channel of the TV program). Note that, to save disk space, the lists contain just the top-3 items for each user, while the available items are five. The algorithm also takes as input the relevance lists of the universal context* `c̃`*, but for brevity we omit to specify them because in this example they are not used. We use the influence values computed in Example 3.1.*

At Lines 2-4, the algorithm determines that the context-aware preferences and influences can be used in the computation, since both Bob and Mark provided feedback during `daytime`*.*

Lines 5-6 perform the necessary initializations: topkMinHeap is set to the empty min-heap, processedItemsSet to the empty set.

Then, the while loop at Line 7 is executed. At the first iteration, the items in the first position of the relevance lists, p_{22} and p_{24}, are considered. p_{22} is added to processedItemsSet, then its group score

is computed as 0.650 and the pair $< p_{22}, 0.650 >$ *is added to the min-heap. Then, also* p_{24} *is added to processedItemsSet and its group score is computed as 0.525; since the min-heap is full (we need to recommend just one item, and it actually contains one item) and the group score of* p_{24} *is lower than that of the item currently in the min-heap,* p_{24} *is not added to the min-heap. Line 19 computes the threshold as* $\frac{1 \cdot 0.8 + 0.333 \cdot 0.9}{1 + 0.333} = 0.825$*. Since the lowest (and only) score in the min-heap is lower than the threshold, the algorithm proceeds with the next iteration.*

At the second iteration, the items at the second position of the relevance lists, p_{24} *and* p_{23}*, are analyzed.* p_{24} *is already in processedItemsSet, so it is not considered any longer.* p_{23}*, on the contrary, is added to processedItemsSet and its group score is computed as 0.375; since the min-heap is full and the score of* p_{23} *is lower than that of the item in the min-heap,* p_{23} *is not added to the min-heap. Line 19 computes the threshold as* $\frac{1 \cdot 0.4 + 0.333 \cdot 0.6}{1 + 0.333} = 0.450$*. Since the lowest (and only) score in the min-heap is greater than the threshold, the algorithm stops and retrieves the item* p_{22} *as the required recommendation.*

Note that in this very simple example only three out of five items have been processed to derive the required recommendation. Note also that the algorithm – to save disk space – has received as input just the upper 3/5 of the relevance lists; still, it has been possible to compute the required recommendation without processing the items excluded from the relevance lists.

5. Experiments

We evaluated our group recommender system from two perspectives: accuracy and efficiency. The section begins with a summary of the main results (Subsection 5.1), and proceeds with the discussion of accuracy (Subsection 5.2) and efficiency (Subsection 5.3).

5.1. Summary of the Evaluation

The experiments assess the quality of the group recommendation technique proposed in this paper with respect to both accuracy and efficiency.

The accuracy is evaluated using a dataset in the TV domain, a domain particularly suitable to carry out this appraisal, because we are interested in new items and in such a domain most of the items are new. The experiments prove that our strategy outperforms the existing approaches in terms of recall and, in more detail, the performance gain grows:

- When the recommendation list is shorter, because many users watch just a limited number of channels, and therefore even simple strategies are able to identify the proper program in lists containing a large

number of items.

- When the group size is larger, because in large groups social factors are more important.
- When the group is more heterogeneous, because if the group is homogeneous the members have the same tastes, and therefore it is not important who are the most influencing ones.

The experiments show also the relevance of employing the context in the influence computation, by comparing the method proposed in this paper with a variant of itself adopting non-contextual influences. Moreover, the significance of the improvements is confirmed through a statistical test.

The efficiency of computing top-K recommendation lists online is assessed by determining the execution time and the number of items to be processed to provide online top-K recommendations, relying on the same TV dataset of the accuracy evaluation plus an additional partially synthetic one derived from Last.fm. The experiments show that our TA-based algorithm can substantially speed up the recommendation process, thus allowing a more fluid interaction with the recommender system. In addition, the experiments show that for the employed datasets the length of the relevance lists might be significantly reduced, saving disk space, without affecting the efficiency.

5.2. Accuracy Evaluation and Discussion

The dataset from the TV domain we used for accuracy evaluation contains implicit feedback as well as context information. Actually, the TV scenario is a perfect setting for evaluating our approach: first, watching TV is a typical group activity, and, second, many kinds of TV programs – like the movies – are broadcast once and then not any longer, at least for a long time, and this makes the "new item" problem extremely relevant.

Do note that the dataset we employed contains implicit feedback obtained from the observation of real groups constituted by *people really watching TV together*, whereas several approaches from the literature (e.g., [39, 6]) use datasets containing just individual feedback (like Movielens), but the groups are built "artificially".

As a consequence, by using this dataset, we have been able to simulate a very large user study, using a test set composed of real people: our system produced the recommendations of the TV programs to groups of the test set who actually watched TV together, and these recommendations were checked against the real choices of that group in that specific moment. If the recommended item was the one that had been actually chosen by the group, we considered the users satisfied by the recommendation.

This is tantamount to conducting a very rich user study where the recruited group members really discuss with each other about the TV program to be watched together, but we have many more users and logged interactions than those usually available in user studies: the experiment dataset contains almost 8 thousand users, while the test set contains a time period of about 15 days – i.e., more than two hundred thousand entries.

We actually found two more (publicly available) datasets containing feedback by real groups: Meetup, adopted in [34], and Plancast, employed in [51]. Unfortunately, they both contain just group feedback, and this makes them unsuitable to test our methodology, where the computation of the influence needs a comparison between the behavior of a user when alone and when in groups, and therefore, besides group feedback, it requires individual feedback as well.

Note that another, publicly released TV dataset[32] was built through a user study and contains individual feedback: each log entry indicates the item chosen by an individual user and the context. The context information specifies also whether the user was alone or in a group when choosing the item, thus making it possible to distinguish the group viewings from the individual ones, but the identity of the possible other group members is not known and in most cases they did not even participate in the user study. Since the feedback from the other group members is not available their preferences and influences cannot be computed; this makes our recommendation algorithm not applicable, and therefore this dataset cannot be employed to check its accuracy.

The preference functions adopted to determine the contextual and non-contextual scores of the items for the single users in the TV dataset are those recently proposed in [15], a state-of-the-art technique to generate TV program recommendations for individual users relying on implicit feedback. It is a content-based method, thus allows to compute users' preference values also for new items.

Note again that, as underlined in the running example, in this special domain all the recommendation lists are generated, rather than as subsets of the whole set of items (i.e., available TV programs), just as subsets of the programs aired at the time instant in which the recommendation is required.

5.2.1. Dataset

The employed dataset contains TV viewing information related to 7,921 users and 119 channels, broadcast both over the air and by satellite. The dataset is composed of an electronic program guide (EPG) containing the description of 21,194 distinct programs, and a log containing both individual and group viewings performed by the users. The attributes available for each program in the EPG are its genre and the channel on which it is transmitted.

Table 2: Time slots

Start time	End time	Description
02:00:00	07:00:00	Graveyard slot
07:00:00	09:00:00	Early morning
09:00:00	12:00:00	Morning
12:00:00	15:00:00	Daytime
15:00:00	18:00:00	Early fringe
18:00:00	20:30:00	Prime access
20:30:00	22:30:00	Prime time
22:30:00	02:00:00	Late fringe

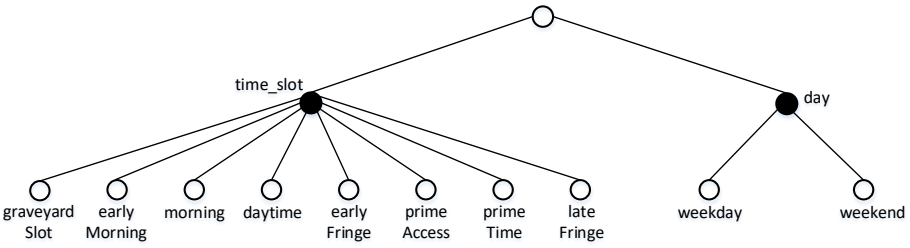

Figure 5: Tree-based representation of the context dimensions available in the experiments

The log spans from December 3rd, 2012 to March 1st, 2013 and contains 4,968,231 entries, among which we retained just the syntonizations longer than three minutes. 3,519,167 viewings are performed by individual users, and are those used to feed the methodology [15] for the computation of the individual preferences of the group members. The remaining 1,449,064 viewings have been done by more than one person. Each log row specifies the identifier of the user and that of the program watched, along with start time and end time. Start and end times were employed to derive the day of the week (weekday vs. weekend) and the time slot, which are the context dimensions considered in the experiments; the available values for the time slot are shown in Table 2, while Figure 5 provides a tree-based representation of the context dimensions and their values.

The group viewings were split into a training set, including the syntonizations between December 3rd, 2012 and February 15th, 2013 (1,210,316 entries), and a test set, containing the remaining ones (238,748 entries). The former was used as the log $\mathcal{L}$ in input to the formulas proposed in this paper, while the latter was employed to assess the quality of the obtained recommendations.

5.2.2. Evaluation Metrics

We evaluated the performance of our recommendation algorithm using *Recall@K*, a value that describes the number of test items included in recommendation lists of length K, computed at the instant in which the viewing of the test items started and in the context in which they were watched.

More formally, let v be a program view in the test set, v_t the start time of the view, v_G the group of users that watched the program, v_i the program watched and v_c the context in which the view took place. $TopK(G, c, t)$ indicates the set of top-K items for the group G in context c among those on the air at time instant t, determined using the recommendation methodology to be evaluated. Recall@K is computed as follows:

$$Recall@K = \frac{|v \in \text{Test Set} : v_i \in TopK(v_G, v_c, v_t)|}{|v \in \text{Test Set}|} \tag{5}$$

We executed experiments for $K = 1$, $K = 2$ and $K = 3$.

5.2.3. Compared Methods

The method proposed by this paper, dubbed *CtxInfl*, was compared with several other techniques, among those presented in Section 2 that allow to recommend new items to ephemeral groups:

- Average (AVG)
- Least misery (LM)
- Maximum satisfaction (MS)
- The approach of [4], taking into account the disagreement between the group members (DIS)
- The GreedyVar algorithm proposed in [33], exploiting the concept of fairness (FAI)
- Weighted average using the expertise, where the expertise of a user u in our experiments is defined as the number of program viewings performed by u (EXP)

In addition, our methodology was compared with a version of itself exploiting only non-contextual influences, i.e., computing influences always using the universal context $\tilde{c}$ in Equation (2); this approach is named *Infl*. Such a comparison is important to verify whether using the context to model the influence may actually provide a performance gain.

Finally, we compared CtxInfl with its CtxInflMix variant described in Section 3.

Besides the evaluation on the whole test set, we also executed experiments considering different subsets of the test items on the basis of group size, heterogeneity and demographic features . About the group sizes, we measured the recall related to the groups containing two (192,378 test items), three (35,216 test items), four (9,739 test items) and five (1,151 test items) members. The heterogeneity of the group was measured using the Spearman's footrule distance with location parameter ℓ [22] between top-K lists[10], employing

[10]The Spearman's footrule distance is the L_1 distance between two permutations of the elements of a domain. Let τ_1 and τ_2 be two permutations of the elements of a domain D, and let τ_1' and τ_2' be the functions assigning the position of each element

$K = 3$ and $\ell = K + 1$. The distance was measured between the top-3 lists of favorite items of all the pairs of group members and then averaged. The values of this distance range from 0 if the top-3 lists of the users contain the same items identically ordered, to 12 if the top-3 lists of the users are completely disjoint. We measured the recall related to the groups with average distance h such that $h \leq 4$ (46,935 test items), $4 < h \leq 8$ (99,333 test items), and $h > 8$ (92,480 test items). Regarding the demographic features, we considered first the gender: groups composed only of adult females (15,921 test items), adult males (4,953 test items), children aged less than or equal to 14, whose gender is not specified in our data (2,991 test items). In addition, we subset the test items on the basis of the age slot: groups constituted only by people aged between 15 and 30 (1,718 test items), between 31 and 50 (23,120 test items), and more than 50 (123,919 test items).

Moreover recall that, in Section 3, we explained that CtxInfl uses non-contextual preferences and influences (i.e., those associated with the universal context $\tilde{\mathtt{c}}$) for the groups containing at least one member whose feedback in the context of interest is unknown. As a consequence, CtxInfl and Infl apply exactly the same formulas to all the test items v such that at least one user in v_G provided no feedback in the context v_c, thus making scarcely significant the comparison of CtxInfl and Infl on these items. Therefore, to better analyze the relevance of the context in influence modeling we compared CtxInfl and Infl also on further restrictions of the test sets listed above, containing just the test items v such that all the users in v_G provided their feedbacks in the context v_c. Each test set obtained through this further restriction contains about 90% of the original test items.

Conversely, CtxInfl and CtxInflMix behave in the same way on all the test items v such that all the users in v_G provided feedback in the context v_c, thus to better compare these two strategies we ran them on the restrictions of the test sets above obtained by reducing them to the test items v such that at least one user in v_G provided no feedback in the context v_c. Each test set obtained in this way contains about 10% of the test items of the original ones.

5.2.4. Results

In this subsection we present the results obtained in the experiments. Tables 3, 4 and 5 show, respectively, Recall@1, Recall@2 and Recall@3, expressed in percentage, measured on the whole test set and on the test

---

of D respectively in the permutations τ_1 and τ_2. The Spearman's footrule distance between τ_1 and τ_2 is defined as follows:

$$F(\tau_1, \tau_2) = \sum_{i \in D} |\tau_1'(i) - \tau_2'(i)|$$

The Spearman's footrule distance with location parameter ℓ is a variation allowing to deal with top-K lists, where all the elements of the domain not included in the top-K list are placed by default at position ℓ.

Table 3: Recall@1

Test set	**CtxInfl**	**CtxInflMix**	**Infl**	**AVG**	**LM**	**MS**	**DIS**	**FAI**	**EXP**
All	37.83%	37.54%	36.09%	33.24%	30.35%	31.91%	33.57%	32.45%	34.56%
Group size = 2	38.36%	38.13%	36.75%	34.18%	31.71%	33.21%	34.47%	33.44%	35.52%
Group size = 3	35.42%	34.92%	33.37%	29.54%	25.87%	27.04%	30.10%	28.64%	30.54%
Group size = 4	36.58%	35.85%	33.54%	28.80%	21.27%	24.78%	29.41%	27.40%	31.34%
Group size = 5	36.40%	37.01%	33.28%	30.76%	19.55%	26.24%	30.84%	27.19%	28.32%
$h \leq 4$	46.98%	46.78%	45.91%	43.07%	42.71%	42.40%	43.16%	42.97%	44.33%
$4 < h \leq 8$	39.90%	39.60%	38.16%	35.38%	33.01%	33.85%	35.79%	34.61%	36.88%
$h > 8$	30.96%	30.65%	28.87%	25.95%	21.21%	24.50%	26.30%	24.77%	27.12%
Females	39.16%	39.24%	38.26%	35.15%	32.45%	34.55%	35.31%	34.71%	37.35%
Males	37.43%	37.23%	36.34%	34.34%	31.70%	32.83%	34.85%	32.26%	34.67%
Children 0-14	32.23%	31.73%	30.39%	30.26%	28.22%	29.25%	30.39%	27.92%	31.06%
$14 < \text{age} \leq 30$	32.36%	32.25%	31.32%	29.34%	28.99%	29.45%	29.10%	28.87%	31.26%
$30 < \text{age} \leq 50$	30.44%	30.20%	29.01%	27.66%	25.95%	26.57%	27.91%	26.25%	28.14%
age > 50	41.31%	41.01%	39.53%	36.52%	33.82%	35.44%	36.91%	36.44%	37.96%

Table 4: Recall@2

Test set	**CtxInfl**	**CtxInflMix**	**Infl**	**AVG**	**LM**	**MS**	**DIS**	**FAI**	**EXP**
All	54.09%	53.87%	53.25%	51.61%	47.10%	50.56%	51.94%	50.04%	52.48%
Group size = 2	54.80%	54.59%	54.15%	52.84%	48.62%	51.98%	53.12%	51.16%	53.67%
Group size = 3	51.48%	51.27%	50.09%	47.50%	41.51%	45.76%	47.92%	45.79%	48.00%
Group size = 4	50.40%	49.84%	48.24%	43.54%	38.26%	41.28%	44.56%	44.38%	47.08%
Group size = 5	49.52%	49.43%	47.61%	46.66%	43.44%	43.44%	47.00%	44.83%	43.87%
$h \leq 4$	65.70%	65.54%	65.16%	64.47%	62.91%	63.82%	64.43%	62.02%	64.87%
$4 < h \leq 8$	56.68%	56.52%	55.99%	54.32%	50.18%	53.30%	54.70%	53.02%	55.45%
$h > 8$	45.40%	45.09%	44.27%	42.18%	35.77%	40.89%	42.65%	40.77%	43.03%
Females	56.27%	56.16%	56.28%	54.05%	49.08%	53.52%	54.31%	52.75%	56.12%
Males	52.74%	52.51%	51.50%	51.46%	46.25%	50.41%	51.65%	51.02%	51.36%
Children 0-14	47.01%	47.04%	48.08%	48.01%	44.93%	43.70%	48.75%	43.53%	49.11%
$14 < \text{age} \leq 30$	45.17%	44.94%	46.16%	46.10%	45.17%	45.98%	46.57%	45.52%	44.88%
$30 < \text{age} \leq 50$	44.81%	44.47%	43.98%	42.93%	40.85%	42.37%	43.60%	42.35%	43.41%
age > 50	58.56%	58.36%	57.78%	56.36%	51.91%	55.59%	56.58%	54.44%	57.33%

sets restricted, as explained above, on the basis of group size, heterogeneity and demographic features. Table 6, on the other hand, compares only CtxInfl and Infl showing Recall@1, Recall@2 and Recall@3 on the test sets further restricted by removing the test items v such that at least one user in v_G did not provide feedback in the context v_c. Table 7, finally, shows Recall@1, Recall@2 and Recall@3 for CtxInfl and CtxInflMix on the test sets restricted by removing the test items v such that all the users in v_G provided feedback in the context v_c. The tables report on the rows the test set composition, and on the columns the employed technique. Our algorithm CtxInfl and its variants CtxInflMix and Infl were run setting $K = 1$ in Equation (2), since this is the value achieving the best results.

Table 5: Recall@3

Test set	**CtxInfl**	**CtxInflMix**	**Infl**	**AVG**	**LM**	**MS**	**DIS**	**FAI**	**EXP**
All	64.27%	64.11%	64.06%	62.94%	58.26%	61.82%	63.25%	60.63%	63.43%
Group size = 2	65.11%	64.92%	64.99%	64.10%	59.71%	63.12%	64.34%	61.74%	64.48%
Group size = 3	61.18%	61.15%	60.76%	59.19%	52.82%	57.54%	59.81%	56.71%	59.51%
Group size = 4	60.12%	59.90%	59.06%	55.42%	50.48%	53.26%	55.93%	54.11%	57.91%
Group size = 5	59.69%	59.08%	58.73%	54.30%	55.60%	54.30%	54.21%	54.74%	59.08%
$h \leq 4$	75.69%	75.60%	75.70%	74.98%	73.11%	74.32%	75.30%	71.28%	75.33%
$4 < h \leq 8$	67.02%	66.91%	66.93%	65.96%	61.70%	64.95%	66.24%	63.96%	66.40%
$h > 8$	55.53%	55.26%	55.07%	53.59%	47.04%	52.13%	53.92%	51.64%	54.19%
Females	66.86%	66.60%	66.90%	65.65%	60.83%	64.95%	65.86%	64.12%	66.74%
Males	61.62%	61.68%	62.02%	62.04%	56.94%	61.01%	62.10%	60.89%	61.80%
Children 0-14	59.08%	59.21%	59.14%	59.38%	55.03%	54.86%	59.08%	54.60%	59.61%
$14 < \text{age} \leq 30$	54.42%	54.37%	54.54%	55.01%	53.90%	53.55%	55.01%	53.26%	54.31%
$30 < \text{age} \leq 50$	54.47%	54.18%	54.34%	53.79%	51.17%	53.10%	54.17%	52.24%	53.44%
age > 50	69.12%	68.98%	68.97%	67.93%	63.41%	66.97%	68.22%	65.21%	68.51%

Table 6: Recall measured considering only the test items v such that all the users in v_G provided feedback in the context v_c

Test set	**Recall@1**		**Recall@2**		**Recall@3**	
	CtxInfl	**Infl**	**CtxInfl**	**Infl**	**CtxInfl**	**Infl**
All	39.07%	37.12%	55.56%	54.62%	65.74%	65.51%
Group size = 2	39.51%	37.72%	56.18%	55.45%	66.47%	66.34%
Group size = 3	36.96%	34.60%	53.36%	51.76%	63.14%	62.66%
Group size = 4	38.33%	34.79%	51.90%	49.40%	61.64%	60.42%
Group size = 5	37.66%	34.07%	50.55%	48.35%	59.84%	58.74%
$h \leq 4$	48.61%	47.40%	67.83%	67.21%	77.76%	77.76%
$4 < h \leq 8$	40.90%	38.98%	57.95%	57.18%	68.25%	68.15%
$h > 8$	32.25%	29.90%	46.74%	45.47%	56.94%	56.43%
Females	40.04%	39.06%	57.14%	57.16%	67.43%	67.47%
Males	39.77%	38.52%	55.38%	53.97%	63.98%	64.44%
Children 0-14	33.58%	31.52%	48.65%	49.85%	60.59%	60.67%
$14 < \text{age} \leq 30$	34.18%	33.01%	46.60%	47.71%	55.75%	55.88%
$30 < \text{age} \leq 50$	31.81%	30.20%	46.41%	45.48%	56.08%	55.93%
age > 50	42.43%	40.44%	59.91%	59.04%	70.48%	70.32%

Table 7: Recall measured considering only the test items v such that at least one user in v_G provided no feedback in the context v_c

Test set	**Recall@1**		**Recall@2**		**Recall@3**	
	CtxInfl	**CtxInflMix**	**CtxInfl**	**CtxInflMix**	**CtxInfl**	**CtxInflMix**
All	27.37%	24.72%	41.73%	39.67%	51.91%	50.36%
Group size = 2	28.07%	25.79%	42.51%	40.44%	52.93%	51.07%
Group size = 3	24.91%	21.01%	38.65%	37.05%	47.79%	47.59%
Group size = 4	25.84%	20.63%	41.11%	37.15%	50.73%	49.19%
Group size = 5	28.00%	32.67%	42.67%	42.00%	58.67%	54.00%
$h \leq 4$	34.60%	32.88%	49.62%	48.19%	60.07%	59.30%
$4 < h \leq 8$	30.42%	27.37%	44.77%	43.07%	55.45%	54.33%
$h > 8$	20.76%	17.99%	34.80%	32.05%	44.37%	42.01%
Females	28.49%	29.57%	45.51%	44.10%	59.88%	56.48%
Males	21.34%	19.75%	34.55%	32.80%	45.38%	45.86%
Children 0-14	21.28%	16.72%	33.74%	34.04%	46.81%	48.02%
$14 < \text{age} \leq 30$	17.55%	16.49%	33.51%	31.38%	43.62%	43.09%
$30 < \text{age} \leq 50$	19.46%	17.32%	31.98%	28.90%	41.61%	39.00%
age > 50	31.32%	28.31%	46.47%	44.46%	56.87%	55.51%

5.2.5. Result Analysis

In the following we analyze in detail the results reported in the tables, focusing first on the comparison of CtxInfl with the competitors from the literature, and then examining the differences between CtxInfl and its non-contextual variant. Finally, CtxInfl is compared with CtxInflMix.

Comparison with the Competitors from the Literature. The comparison starts with the whole test set, and then considers the results obtained when changing the group size and the group heterogeneity (as defined in Subsection 5.2.3).

Full Test Set The recall values we obtained highlight that our strategy (CtxInfl) dominates its competitors for all the experimented lengths of the recommendation list. Also notice that, among the other approaches, the best-performing one is EXP, and this suggests the goodness of the strategies relying on a weighted average with respect to plain average, least misery, maximum satisfaction, and also to the approach of [4], taking disagreement into account. Another aspect that can be noticed from the results is that, with this dataset, the difference between our method and the baselines is larger when recommending few items: 3.27% for Recall@1, 1.61% for Recall@2 and 0.84% for Recall@3. This happens because many users watch just a limited number of channels, and therefore even simple strategies are able to identify the proper program in lists containing a larger number of items.

Results when Varying the Group Size In these experiments CtxInfl is also the best one, and its main competitor in most cases remains EXP. It is immediately evident that the advantage obtained by CtxInfl with respect to the best alternative grows when the group size is bigger: considering Recall@1, the difference is 2.84% when the group size is 2, 4.88% when the group size is 3, 5.24% when the group size is 4, and 5.56% when the group size is 5. This result is easily interpreted: when groups are larger social factors are more important.

Results when Varying the Group Heterogeneity Once again, CtxInfl is the leading technique and EXP the second one. The first evidence is that lower heterogeneity corresponds to higher recall; therefore, as it is intuitive, recommending to homogeneous groups is a simpler problem. Moreover, the gain brought by CtxInfl with respect to EXP grows with the heterogeneity of the groups: considering Recall@1, the gain is 2.65% for $h \leq 4$, 3.02% for $4 < h \leq 8$ and 3.84% for $h > 8$. Also this outcome is very intuitive: if the group is homogeneous the members have the same tastes, and therefore it is not important who are the most influencing ones.

Results when Varying the Group Demographic Features CtxInfl is the best performing technique in terms of Recall@1, ahead of EXP. For instance, the difference between the two techniques is 2.77%

for male adults, and 3.35% for groups composed of people being more than 50 years old. However, when longer recommendation lists are considered, the differences between the alternatives decrease and in some cases CtxInfl is also slightly overtaken by EXP or AVG. The good performance shown by AVG suggests that when groups are demographically homogeneous it is less necessary to employ very sophisticated algorithms, at least when recommendation lists are not too short. In addition, note that the absolute values of the recall are quite lower for children than they are for adult females and males, indicating that children may be more difficult to satisfy. Finally, we also observe that the recall values measured for the groups constituted only by people being more than 50 years old are very high; this may be explained by the fact that these users are probably more habit-bound, and so it is easier to find appropriate items to be recommended.

Relevance of the Context in Influence Computation. The results clearly show that adding the context to the influence improves the performance for all the experimented test sets. The improvements brought by CtxInfl with respect to Infl are more evident in the results shown in Table 6 than in those shown in Tables 3, 4 and 5, because the former consider just the test items associated with contexts in which the two methods apply different formulas; in fact, when determining recommendations in contexts where at least one group member did not provide any feedback, both CtxInfl and Infl compute the scores using in Equation (3) the universal context $\tilde{c}$. In more detail, in Table 6 we observe that the gain seems to be bigger when the recommendation lists are shorter: considering all the test items v such that all the members of v_G provided feedback in the context v_c, the benefit provided by the context is 1.95% for Recall@1, 0.94% for Recall@2 and 0.23% for Recall@3. Also bigger groups seem to allow greater performance improvements: for Recall@1, the gain is 1.79% when the group size is 2, 2.35% when the group size is 3, 3.54% when the group size is 4, and 3.59% when the group size is 5. Moreover, the performance improvement grows with the group heterogeneity: considering Recall@1, the gain is 1.21% for $h \leq 4$, 1.92% for $4 < h \leq 8$ and 2.35% for $h > 8$. Finally, the latter observation is confirmed when the heterogeneity is demographic: when the group members share common demographic features, the difference between the two variants is low. For instance, considering Recall@1, the gain is only 0.98% for groups constituted just by adult females, and only 1.18% for groups containing only people being between 14 and 30 years old.

Comparison between CtxInfl and CtxInflMix. Finally, to compare CtxInfl with CtxInflMix, we analyze Table 7 that considers just the test items on which the two strategies behave differently. As anticipated in Section 3, CtxInfl shows a better accuracy for almost all the experimented test sets and lengths of the recommendation list. To understand why CtxInflMix, which "blends" values measured on different subsets of the log, showed a

Table 8: Values of the McNemar test statistic

	CtxInflMix	**Infl**	**AVG**	**LM**	**MS**	**DIS**	**FAI**	**EXP**
Recall@1	178.8	1,298.5	5,925.1	6,501.6	8,299.0	4,673.1	5,563.5	2,953.3
Recall@2	107.8	311.3	2,152.9	5,111.6	3,259.2	1,428.7	3,699.0	778.1
Recall@3	65.3	24.1	727.5	3,897.5	1,721.9	387.9	3,504.0	262.9

bad behavior in the experiments we did a manual check, sampling users from our dataset and controlling how much their influence varies according to context, and how far these values are from their non-contextual influence. Often, for the same user, non-contextual influence turned out to be quite distant from the contextual one, thus we explain this somehow unexpected behavior with the fact that putting together contextual and non-contextual influences creates a sort of "unbalanced situation" among the members of the group, which reflects reality worse than keeping the weights uniform.

5.2.6. Accuracy Improvement Validation

To validate the significance of the accuracy improvement provided by our algorithm, a statistical test was used. The statistical test is employed to verify that the performance differences between two approaches are significant and not just due to chance; in more detail, the aim is to reject the null hypothesis stating that the compared methods have the same performance. Our data are split into training and test set on a temporal basis and thus we can have just a single training set/test set pair, therefore we compared CtxInfl with the other algorithms using McNemar's statistical test, which is appropriate in this situation [19].

McNemar's test defines a test statistic on the basis of the numbers of test items that are classified correctly or not correctly by the two compared methods. The greater is the value of the statistic, the lower is the p-value of the test and therefore more evidence is provided to reject the null hypothesis of no improvement. In practice, the null hypothesis is usually rejected with p-values lower than 0.05 or 0.01 [26], which correspond to values of the McNemar test statistic greater than 3.8415 or 6.6349, respectively.

McNemar's test was executed considering the whole test set for Recall@1, Recall@2 and Recall@3. Table 8 shows the values of the McNemar test statistic obtained by comparing CtxInfl with its competitors. The measured values are very high, corresponding to p-values very close to 0 and thus providing strong evidence for rejecting the hypothesis of no improvement.

5.3. Efficiency of the Online Top-K Recommendation Task

To assess the efficiency of our strategy in providing online top-K recommendations, we executed experiments on the previous dataset and on another partially synthetic one, measuring the execution time and the number of processed items; please notice that the additional partially synthetic dataset could not be used

in the accuracy evaluation, since it does not contain feedback from real groups. All the experiments were performed on a quad-core 2.80 GHz Intel Core i7 machine with 16 GB of RAM, running Windows 10.

5.3.1. Dataset

The experiments evaluating the efficiency were executed on two datasets: the TV dataset already described in Section 5.2.1, and an additional (partially synthetic) dataset built from *Last.fm* [9].

The original Last.fm dataset contains 1,019,318 users, 361,533 items (i.e., songs), and a log with 48,373,586 listenings performed by individual users; the content of each song is described by means of tags. In our experiments we considered 2,000 users, all the items, the whole log, and the 300 most common tags. The dataset was then enriched with the user influences, randomly generated with uniform distribution in the interval [0, 1]. A synthetic log – to be used as test set – was generated as well, envisaging 2,000 group listenings; each group in this log contains exactly three users, and each user has the same probability to be included in a group. The context, not available in the dataset, was not synthetically generated: under the efficiency point of view, indeed, adding the context would have just increased the dataset size, which we deemed big enough. The individual preferences of the users for the items were computed in a content-based way exploiting the preferences for the tags, which were in their turn derived on the basis of the log of listenings performed by the individual users.

Do notice that for both the datasets it is not required to store all the very numerous preferences of the individual users for the items. Indeed, it suffices to store the preferences for the relatively few values of the item attributes (e.g., tags, channels, genres, ...), and then use these preferences to easily derive those referred to the items.

5.3.2. Experiments and Metrics

For each of the two datasets we computed the top-K recommendations for all the groups associated with the viewings/listenings in the test sets. Then, we measured:

- The average number of items processed to compute the top-K recommendations
- The average execution time to compute the top-K recommendations

Since we are testing efficiency, we adopt a "harder version" of the TV dataset problem where all the 18,731 TV programs are supposed to be available to be watched in every time instant. The experiments were repeated for various values of K, the size of the recommendation lists: 3, 20, 100, 500, 5,000. In addition, we considered to have different storage possibilities for the portions of the relevance lists: 100%,

Table 9: TV dataset: average number of processed items and average execution time

	K=3	**K=20**	**K=100**	**K=500**	**K=5,000**
100%	236.58 0.54 ms	257.49 0.56 ms	357.63 0.80 ms	910.96 2.17 ms	5,699.38 15.95 ms
80%	236.58 0.51 ms	257.49 0.55 ms	357.63 0.78 ms	910.96 2.08 ms	5,050.44 11.87 ms
60%	236.58 0.53 ms	257.49 0.55 ms	357.63 0.78 ms	910.96 2.06 ms	5,050.44 12.09 ms
40%	236.58 0.53 ms	257.49 0.55 ms	357.63 0.78 ms	910.96 2.06 ms	5,050.44 12.20 ms
20%	236.58 0.53 ms	257.49 0.55 ms	357.63 0.78 ms	910.96 2.06 ms	14,653.96 45.48 ms
1%	5,479.73 20.04 ms	5,528.40 18.90 ms	5,782.91 20.57 ms	18,724.00 68.36 ms	18,731 66.50 ms
0%	18,731 60.24 ms	18,731 61.55 ms	18,731 63.10 ms	18,731 65.46 ms	18,731 62.82 ms

Table 10: Modified Last.fm dataset: average number of processed items and average execution time

	K=3	**K=20**	**K=100**	**K=500**	**K=5,000**
100%	1,665.98 6.68 ms	2,681.14 9.42 ms	4,123.55 13.68 ms	7,252.89 24.70 ms	21,951.93 84.17 ms
80%	1,665.98 6.37 ms	2,681.14 9.47 ms	4,123.55 13.65 ms	7,252.89 24.66 ms	21,951.93 80.77 ms
60%	1,665.98 6.25 ms	2,681.14 9.78 ms	4,123.55 14.00 ms	7,252.89 25.33 ms	21,951.93 81.66 ms
40%	1,665.98 5.44 ms	2,681.14 8.89 ms	4,123.55 15.12 ms	7,252.89 26.10 ms	21,951.93 86.64 ms
20%	1,665.98 5.44 ms	2,681.14 8.94 ms	4,123.55 15.60 ms	7,252.89 25.90 ms	21,951.93 83.92 ms
1%	11,210.65 57.95 ms	23,133.19 121.15 ms	39,541.07 208.22 ms	85,838.70 469.27 ms	361,533 1,993.26 ms
0%	361,533 1,274.02 ms	361,533 1,274.08 ms	361,533 1,270.74 ms	361,533 1,309.78 ms	361,533 1,342.91 ms

80%, 60%, 40%, 20%, 1%, 0%. If the relevance lists contain no items (0%), then, to compute the top-K recommendations, it is always necessary to process all the items, without exploiting the TA thresholding principle.

5.3.3. Results

The results for the two datasets are summarized in Tables 9 and 10. The tables rows report the lengths of the relevance lists and the columns the values of K. The cells contain both the average number of items processed and the average execution time in milliseconds.

5.3.4. Result Analysis

The tables highlight how useful the TA-based algorithm is to speed up the online computation of top-K group recommendations; indeed, if the relevance lists are available, the number of processed items decreases

significantly. For example, when considering $K = 3$, on the average, for the TV dataset we need to process 236.58 items instead of 18,731, and for the modified Last.fm dataset 1,665.98 instead of 361,533.

Regarding the execution times, the values for the TV dataset are always quite small (lower than 100 ms) since the total number of available items is limited; as a consequence, for this dataset the reduction in terms of processed items would not produce a significant improvement in the responsiveness of the system. The situation is quite different for the modified Last.fm, which is much bigger: the execution time for the online computation of top-K recommendation lists decreases significantly with the usage of our TA-based algorithm, passing from 1.2 seconds to few milliseconds. This can greatly improve the responsiveness of the system and the fluidity of the interaction. Note that in a real application it would be very annoying for the user to wait more than 1 second to obtain the required information.

Let us now consider the length of the relevance lists. The tables show that, for both datasets and for all the experimented values of K, the length of the relevance lists can be reduced – thus saving disk space – up to 40% with no significant impact either on the number of items to be processed or on the execution time; in most cases the same holds also for 20% of the items. Note that if K is small even having relevance lists containing just 1% of the items allows significant performance gains with respect to the case where no relevance lists are available. This happens because the experimented values of K, as usual in real cases, are actually smaller than the number of items, and therefore the lower part of the relevance lists very often contains items that will not be considered in the top-K computation.

6. Conclusions

This paper has studied a group recommendation methodology *for ephemeral groups*, so that also items that are new in the system can be recommended. This approach derives group preferences by aggregating individual ones, relying on the *contextual influence*, that is, the ability of a user, in a given situation, to inspire the group's decision. Differently from other researches, in our work the influence values are determined exclusively by using log data, without resorting to information solicited from the users.

The effectiveness of the proposed approach has been proven through experiments on a very large dataset in the TV domain, containing the choices of real groups. Moreover, we have proposed an algorithm that allows to efficiently compute the recommendations *at runtime*, evaluated using the same dataset in the TV domain as well as a bigger, partially synthetic dataset.

A possible future work regards a deeper exploration of the benefits of using the notion of context. Our current strategy derives the preferences and influences related to a context c only on the basis of the previous

activities performed in the same context; alternative approaches might also include information about the activities associated with contexts other than c, learned from data or based on some form of similarity (for instance those experienced immediately before c, or those more general than c), using different weights. Other interesting developments deal with the possibility to emphasize aspects like novelty and serendipity when recommending items, and also consider ethical problems like those related to *fairness* and *diversity* [46, 20]. Note that, in some way, these parameters related to ethics have already been used as criteria in the works [4] and [33], discussed in the Related Work section, where fairness was adopted in order to minimize disagreement and take into account all the users. However we think that much work has still to be done to fully understand these, and possibly other *dimensions of ethics* along which the research could be addressed.

Acknowledgments

This research has been partially supported by the IT2Rail project, funded by European Union's Horizon 2020 research and innovation program under grant agreement No: 636078 and by the Italian SHELL project (Cluster Technologies for Living Environments - http://shell.smartlivingtech.it/en/) - CTN01_00128_111357.

References

[1] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, 17(6):734–749, 2005.

[2] G. Adomavicius, R. Sankaranarayanan, S. Sen, and A. Tuzhilin. Incorporating contextual information in recommender systems using a multidimensional approach. *ACM Transactions on Information Systems*, 23(1):103–145, 2005.

[3] I. Ali and S. Kim. Group recommendations: Approaches and evaluation. In *Proc. of IMCOM 2015, 9th International Conference on Ubiquitous Information Management and Communication*, pages 105:1–105:6. ACM, 2015.

[4] S. Amer-Yahia, S. Basu Roy, A. Chawla, G. Das, and C. Yu. Group recommendation: Semantics and efficiency. *PVLDB*, 2(1):754–765, 2009.

[5] S. Amer-Yahia, B. O. Tehrani, S. Basu Roy, and N. Shabib. Group recommendation with temporal affinities. In *Proc. of EDBT 2015, 18th International Conference on Extending Database Technology*, pages 421–432. OpenProceedings.org, 2015.

[6] L. Baltrunas, T. Makcinskas, and F. Ricci. Group recommendations with rank aggregation and collaborative filtering. In *Proc. of RecSys 2010, 4th International Conference on Recommender Systems*, pages 119–126. ACM, 2010.

[7] S. Basu Roy, S. Amer-Yahia, A. Chawla, G. Das, and C. Yu. Space efficiency in group recommendation. *VLDB Journal*, 19(6):877–900, 2010.

[8] S. Berkovsky and J. Freyne. Group-based recipe recommendations: Analysis of data aggregation strategies. In *Proc. of RecSys 2010, 4th International Conference on Recommender Systems*, pages 111–118. ACM, 2010.

[9] T. Bertin-Mahieux, D. P. Ellis, B. Whitman, and P. Lamere. The million song dataset. In *Proc. of ISMIR 2011, 12th International Conference on Music Information Retrieval*. University of Miami, 2011.

[10] C. Bolchini, E. Quintarelli, and L. Tanca. CARVE: Context-aware automatic view definition over relational databases. *Information Systems*, 38(1):45–67, 2013.

[11] S. Bourke, K. McCarthy, and B. Smyth. Using social ties in group recommendation. In *Proc. of AICS 2011, 22nd Irish Conference on Artificial Intelligence and Cognitive Science*, 2011.

[12] A. J. Chaney, M. Gartrell, J. M. Hofman, J. Guiver, N. Koenigstein, P. Kohli, and U. Paquet. A large-scale exploration of group viewing patterns. In *Proc. of TVX 2014, 1st International Conference on Interactive Experiences for TV and Online Video*, pages 31–38. ACM, 2014.

[13] Y. Chen, L. Cheng, and C. Chuang. A group recommendation system with consideration of interactions among group members. *Expert Systems with Applications*, 34(3):2082–2090, 2008.

[14] I. A. Christensen and S. N. Schiaffino. Social influence in group recommender systems. *Online Information Review*, 38 (4):524–542, 2014.

[15] P. Cremonesi, P. Modica, R. Pagano, E. Rabosio, and L. Tanca. Personalized and context-aware TV program recommendations based on implicit feedback. In *Proc. of EC-Web 2015, 16th International Conference on Electronic Commerce and Web Technologies*, pages 57–68. Springer, 2015.

[16] L. M. de Campos, J. M. Fernández-Luna, J. F. Huete, and M. A. Rueda-Morales. Group recommending: A methodological approach based on bayesian networks. In *Proc. of ICDE Workshops*, pages 835–844. IEEE Computer Society, 2007.

[17] T. De Pessemier, S. Dooms, and L. Martens. Comparison of group recommendation algorithms. *Multimedia Tools and Applications*, 72(3):2497–2541, 2014.

[18] A. K. Dey. Understanding and using context. *Personal and Ubiquitous Computing*, 5(1):4–7, 2001.

[19] T. G. Dietterich. Approximate statistical test for comparing supervised classification learning algorithms. *Neural Computation*, 10(7):1895–1923, 1998.

[20] M. Drosou, H. V. Jagadish, E. Pitoura, and J. Stoyanovich. Diversity in big data: A review. *Big Data*, 5(2):73–84, 2017.

[21] R. Fagin, A. Lotem, and M. Naor. Optimal aggregation algorithms for middleware. In *Proc. of PODS 2001, 20th Symposium on Principles of Database Systems*, pages 102–113. ACM, 2001.

[22] R. Fagin, R. Kumar, and D. Sivakumar. Comparing top k lists. *SIAM Journal on Discrete Mathematics*, 17(1):134–160, 2003.

[23] M. Gartrell, X. Xing, Q. Lv, A. Beach, R. Han, S. Mishra, and K. Seada. Enhancing group recommendation by incorporating social relationship interactions. In *Proc. of GROUP 2010, International Conference on Supporting Group Work*, pages 97–106. ACM, 2010.

[24] S. Ghazarian and M. Nematbakhsh. Enhancing memory based collaborative filtering for group recommender systems. *Expert Systems with Applications*, 42(7):3801–3812, 2015.

[25] J. Gorla, N. Lathia, S. Robertson, and J. Wang. Probabilistic group recommendation via information matching. In *Proc. of WWW 2013, 22nd International World Wide Web Conference*, pages 495–504. ACM, 2013.

[26] J. Han, M. Kamber, and J. Pei. *Data Mining: Concepts and Techniques, 3rd edition.* Morgan Kaufmann, 2011.

[27] L. Hu, J. Cao, G. Xu, L. Cao, Z. Gu, and W. Cao. Deep modeling of group preferences for group-based recommendation. In *Proc. of AAAI 2014, 28th International Conference on Artificial Intelligence*, pages 1861–1867. AAAI Press, 2014.

[28] H. Kim and A. El-Saddik. A stochastic approach to group recommendations in social media systems. *Information Systems*, 50:76–93, 2015.

[29] N. Kim and J. Lee. Group recommendation system: Focusing on home group user in TV domain. In *Proc. of SCIS 2014,*

7th International Conference on Soft Computing and Intelligent Systems, pages 985–988. IEEE Computer Society, 2014.

[30] M. Kompan and M. Bieliková. Social structure and personality enhanced group recommendation. In *Proc. of UMAP Workshops*. CEUR-WS.org, 2014.

[31] Y. Koren. Factor in the neighbors: Scalable and accurate collaborative filtering. *ACM Transactions on Knowledge Discovery from Data*, 4(1):1:1–1:24, 2010.

[32] M. S. Kristoffersen, S. E. Shepstone, and Z. Tan. A dataset for inferring contextual preferences of users watching TV. In *Proc. of UMAP 2018, 26th Conference on User Modeling, Adaptation and Personalization*, pages 367–368. ACM, 2018.

[33] X. Lin, M. Zhang, Y. Zhang, Z. Gu, Y. Liu, and S. Ma. Fairness-aware group recommendation with pareto-efficiency. In *Proc. of RecSys 2017, 11th International Conference on Recommender Systems*, pages 107–115. ACM, 2017.

[34] X. Liu, Y. Tian, M. Ye, and W. Lee. Exploring personal impact for group recommendation. In *Proc. of CIKM 2012, 21st International Conference on Information and Knowledge Management*, pages 674–683. ACM, 2012.

[35] P. Lops, M. de Gemmis, and G. Semeraro. Content-based recommender systems: State of the art and trends. In F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, editors, *Recommender Systems Handbook*, pages 73–105. Springer, 2011.

[36] J. Masthoff. Group modeling: Selecting a sequence of television items to suit a group of viewers. *User Modeling and User-Adapted Interaction*, 14(1):37–85, 2004.

[37] J. Masthoff and A. Gatt. In pursuit of satisfaction and the prevention of embarrassment: Affective state in group recommender systems. *User Modeling and User-Adapted Interaction*, 16(3-4):281–319, 2006.

[38] E. Ntoutsi, K. Stefanidis, K. Nørvåg, and H. Kriegel. Fast group recommendations by applying user clustering. In *Proc. of ER 2012, 31st International Conference on Conceptual Modeling*, pages 126–140. Springer, 2012.

[39] M. O'Connor, D. Cosley, J. A. Konstan, and J. Riedl. Polylens: A recommender system for groups of user. In *Proc. of ECSCW 2001, 7th European Conference on Computer Supported Cooperative Work*, pages 199–218. Kluwer, 2001.

[40] L. Quijano-Sánchez, J. A. Recio-García, B. Díaz-Agudo, and G. Jiménez-Díaz. Social factors in group recommender systems. *ACM Transactions on Intelligent Systems and Technology*, 4(1):8, 2013.

[41] E. Quintarelli, E. Rabosio, and L. Tanca. Recommending new items to ephemeral groups using contextual user influence. In *Proc. of RecSys 2016, 10th International Conference on Recommender Systems*, pages 285–292. ACM, 2016.

[42] S. Seko, T. Yagi, M. Motegi, and S. Muto. Group recommendation using feature space representing behavioral tendency and power balance among members. In *Proc. of RecSys 2011, 5th International Conference on Recommender Systems*, pages 101–108. ACM, 2011.

[43] C. Senot, D. Kostadinov, M. Bouzid, J. Picault, A. Aghasaryan, and C. Bernier. Analysis of strategies for building group profiles. In *Proc. of UMAP 2010, 18th International Conference on User Modeling, Adaptation, and Personalization*, pages 40–51. Springer, 2010.

[44] C. Shin and W. Woo. Socially aware TV program recommender for multiple viewers. *IEEE Transactions on Consumer Electronics*, 55(2):927–932, 2009.

[45] R. Sotelo, Y. Blanco-Fernández, M. López-Nores, A. Gil-Solla, and J. J. Pazos-Arias. TV program recommendation for groups based on muldimensional tv-anytime classifications. *IEEE Transactions on Consumer Electronics*, 55(1):248–256, 2009.

[46] J. Stoyanovich, S. Abiteboul, and G. Miklau. Data responsibly: Fairness, neutrality and transparency in data analysis. In *Proc. of EDBT 2016, 19th International Conference on Extending Database Technology*, pages 718–719. OpenProceedings.org, 2016.

[47] M. O. van Deventer, J. de Wit, J. Vanattenhoven, and M. Guelbahar. Group recommendation in an hybrid broadcast broadband television context. In *Proc. of UMAP Workshops*. CEUR-WS.org, 2013.

[48] E. Vildjiounaite, V. Kyllönen, T. Hannula, and P. Alahuhta. Unobtrusive dynamic modelling of TV programme preferences in a Finnish household. *Multimedia Systems*, 15(3):143–157, 2009.

[49] M. Ye, X. Liu, and W. Lee. Exploring social influence for recommendation: A generative model approach. In *Proc. of SIGIR 2012, 35th International Conference on Research and Development in Information Retrieval*, pages 671–680. ACM, 2012.

[50] Z. Yu, X. Zhou, Y. Hao, and J. Gu. TV program recommendation for multiple viewers based on user profile merging. *User Modeling and User-Adapted Interaction*, 16(1):63–82, 2006.

[51] Q. Yuan, G. Cong, and C. Lin. COM: A generative model for group recommendation. In *Proc. of KDD 2014, 20th International Conference on Knowledge Discovery and Data Mining*, pages 163–172. ACM, 2014.

[52] H. Zhu, B. A. Huberman, and Y. Luon. To switch or not to switch: Understanding social influence in online choices. In *Proc. of CHI 2012, International Conference on Human Factors in Computing Systems*, pages 2257–2266. ACM, 2012.