Probabilistic Group Recommendation via Information Matching

Jagadeesh Gorla[1], Neal Lathia[2], Stephen Robertson[3], Jun Wang[1]
[1]Department of Computer Science, University College London, UK
[2]Computer Laboratory, University of Cambridge, UK
[3]Microsoft Research Cambridge, UK
{jgorla,junwang}@cs.ucl.ac.uk, neal.lathia@cl.cam.ac.uk, stephenerobertson@hotmail.co.uk

ABSTRACT

Increasingly, web recommender systems face scenarios where they need to serve suggestions to groups of users; for example, when families share e-commerce or movie rental web accounts. Research to date in this domain has proposed two approaches: computing recommendations for the group by merging any members' ratings into a single profile, or computing ranked recommendations for each individual that are then merged via a range of heuristics. In doing so, none of the past approaches reason on the preferences that arise in individuals *when they are members of a group*. In this work, we present a probabilistic framework, based on the notion of information matching, for group recommendation. This model defines *group relevance* as a combination of the item's relevance to each user as an individual and as a member of the group; it can then seamlessly incorporate any group recommendation strategy in order to rank items for a set of individuals. We evaluate the model's efficacy at generating recommendations for both single individuals and groups using the MovieLens and MoviePilot data sets. In both cases, we compare our results with baselines and state-of-the-art collaborative filtering algorithms, and show that the model outperforms all others over a variety of ranking metrics.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Retrieval models

General Terms

Theory, Algorithms

Keywords

Probabilistic Modelling, Group Recommendation

1. INTRODUCTION

In recent years, Collaborative Filtering (CF) algorithms have become the hallmark of web-based recommender systems. These techniques compute personalised recommendations for users by learning from the ratings or interactions that the users create as they engage with the (movie, music, news) content on the system [12, 28]. While traditional research on recommender systems has focused on finding relevant items for *single* users, there is an increasing occurrence of individual web profiles and accounts being shared amongst a group of people. For example, a household of users may share a single movie-rental and recommendation account, or users of mobile recommender systems may be seeking locations (e.g., restaurants) for a group of friends to go to together. Both of these types of scenarios have led to recent research on *group recommendation* [16]. Group recommendation scenarios tend to differ in terms of how preference data about the group is collected. For example, in the case of shared household web accounts, the ratings that the system represents as a single user may actually reflect a number of people's preferences; in the case of a restaurant-recommender system, multiple profiles may need to be considered simultaneously in order to suitably personalise the system's recommendations. Group recommendation systems have been used in various forms across the web to recommend news pages [22], holidays [19], music [8], and both TV programs and movies [1, 21, 29].

The main challenge behind these scenarios has been that of computing recommendations from a potentially diverse set of group members' ratings. Past approaches have tackled this in one of two ways. On the one hand, all group members' ratings can be folded into a single profile, which can then be treated as a unique user that recommendations should be computed for. Naturally, this approach may bias its output away from those group members that have the sparsest profiles. Alternatively, personalised recommendations can be computed for each group member first, and the resulting set of ranked recommendations can be merged into a single list for the group using pre-defined heuristics [3, 16]. This solution assumes that successful group recommendation comes from the strategies used to merge the ranked lists, without modelling the group as a whole or considering how individuals' preferences may differ when they become part of a group.

In this paper, we tackle the problem of group recommendation for households of users sharing a movie-rental account. Households may have a varied number of members and a non-uniform distribution of ratings amongst any members. We present a probabilistic model for group recommendation; this model combines the relevance of items to different group members with the relevance of items to the group as a whole. In doing so, we show that higher quality recommendations can be computed by shifting the focus toward better modelling of group members, rather than merging individual's ranked recommendations. In summary, we make the following contributions:

WWW 2013, May 13–17, 2013, Rio de Janeiro, Brazil.
ACM 978-1-4503-2035-1/13/05.

- We introduce a number of state of the art approaches to CF (Section 2) and describe the information matching model that can seamlessly compute the probability that an item will be relevant to a given user (Section 3.1) by reasoning on the user's *preference* and the item's *appeal*. Furthermore, the information matching approach allows for users to be described both in terms of their ratings as well as any other contextual feature that is available to the system (e.g., their role in the household).

- We then use this model to derive a framework for group recommendation (Section 3.2) that, unlike previous work—which focuses on merging recommendations computed for individual users—uses the principles of information matching in order to compute the probabilities of items' relevance to a group, while taking the entirety of the group into consideration.

- We evaluate (Section 4) the probabilistic model alongside state-of-the-art CF approaches, including popularity based, neighbourhood, and latent factor models using household rating data from MoviePilot[1]. We find that not only do state-of-the-art methods perform very poorly in this domain, but the probabilistic model performs demonstrably better over a range of different evaluation metrics. We further provide non-group, baseline results using the MovieLens dataset to allow our work to be compared to the research literature.

We believe these results will be useful to practitioners whose systems may have shared groups of users, including: movie rental, e-commerce, and web gaming portals. We close in Sections 5 and 6 by discussing the key differences between information matching and CF alongside our future work.

2. BACKGROUND AND RELATED WORK

In this section, we introduce the problem space that we address, as well as describing a number of algorithms that we will compare our proposed model to. Collaborative filtering (CF) [12, 28] is the mainstream approach to building web recommender systems: it is based on the assumption that, as users give ratings for items on a web site, their discovery of new items can be aided by learning from the ratings that other users have given. Broadly speaking, CF uses a matrix of user-item ratings in order to compute predictions for those items that users have not rated; items can then be ranked, in descending order, by these predictions.

In the following sections, we describe a baseline and two state-of-the-art CF algorithms that we will compare against when we evaluate our approach. First, we use a non-personalised, popularity-based, baseline. Next, we compare with two techniques that pervade the literature in the field: neighbourhood approaches (Section 2.2) and latent factor models (Section 2.3). Broadly speaking, neighbourhood approaches are based on the similarity between users or items. The latent factor model, instead, represents the user and item in the same latent space with a predefined number of hidden dimensions. This is generally a lower dimensional space and the rating between the user-item pair is predicted by the proximity of the relevant latent factors. Finally, we discuss how these algorithms have been used in conjunction with other heuristics in order to generate recommendations for groups (Section 2.4), and the methods and challenges related to evaluating group recommendations (Section 2.5).

[1]Released as part of the Context-Aware Movie Recommendation Challenge at ACM RecSys 2011: http://2011.camrachallenge.com/2011

2.1 The Popularity Model

The baseline that we test all methods against is a simple, non-personalised approach based on *popularity*. The popularity of an item is defined as being proportional to the number of ratings that it has received by any user; by counting ratings, we can rank items in descending order of popularity. Although this model is very simple, in the experiments that follow we find that it is in-fact very effective in terms of performance. In the following, we will refer to this baseline as *Pop-item*.

2.2 Neighbourhood Models

The k-Nearest Neighbourhood algorithm has been used in recommender systems research since its inception [10]. There are two different flavours of the algorithm: *user* and *item*-based. The user-based approach represents users as a sparse, high-dimensional vector of item ratings. The item-based alternative, instead, represents items as sparse, high dimensional vectors of user ratings. Both methods operate by measuring any similarity between the users or items in order to predict a score for unrated items. In this work, we use the Apache Mahout library[2] implementation of the user and the item-based methods, which uses the Pearson correlation to measure similarity.

Specifically, the similarity between a user u and v is measured as:

$$sim(u,v) = \frac{\Sigma_{i \in I_{uv}} (r_{u,i} - \hat{r_u})(r_{v,i} - \hat{r_v})}{\sqrt{\Sigma_{i \in I_{uv}} (r_{u,i} - \hat{r_u})^2} \sqrt{\Sigma_{i \in I_{uv}} (r_{v,i} - \hat{r_v})^2}} \quad (1)$$

where I_{uv} is the set of items rated by both users u, v and $\hat{r_u}, \hat{r_v}$ are the average rating of co-rated items of u and v respectively.

Once similarities have been computed, the predicted rating for a user-item pair is computed as the similarity-weighted average of the ratings that (in the item-based approach) the user has given to the k most similar items or (in the user-based approach) the ratings that the k most similar users have given to the item. More formally, the predicted rating $\hat{r}(u,i)$ for user u and item i, in the user-based approach, is computed with:

$$\hat{r}(u,i) = \bar{r}_u + \frac{\Sigma_{j=1}^{k} \left(sim(n_j,u)(r_{n_j,i} - \bar{r_{n_j}}) \right)}{\Sigma_{j=1}^{k} sim(n_j,u)} \quad (2)$$

where $\bar{r}_u$ is the mean rating of user u, and n_j is the jth neighbour of u. In the following experiments, we set the neighbourhood size k to 50 for both approaches.

2.3 Latent Factor Model

Many recent recommender algorithms have been based on latent factor models [17]. The idea behind these models is to factorize the rating matrix into two lower rank matri-

[2]http://mahout.apache.org/

ces, one capturing user factors ($\mathbf{p}_u$), and one for item factors ($\mathbf{q}_i$). Each user and item is represented over a fixed $f-$dimensional feature space, where f is a low-dimensional parameter to the model. A predicted rating for a user-item pair (u, i) is then computed with the inner product between the related factor vectors:

$$\hat{r}(u,i) = \mathbf{p}_u \mathbf{q}_i^T \tag{3}$$

Majority of recent models learn the factor vectors using any available ratings and an objective function; in this work, we follow Koren *et al.* [17] and learn the vectors' factors via a gradient descent objective function. To be consistent with the literature, we will refer to this model as *PureSVD*. It uses $f = 150$ features which are optimised by running over 60 iterations. We use the exact implementation used in [4]. We also reproduced the results reported in [4] using the ranking evaluation strategies to verify the model before we conducted our experimentation.

2.4 Group Recommendation

All the above methods can be used to generate recommendations for an individual user. In this section, we describe the strategies that have been historically implemented in order to produce group recommendations. Broadly speaking, there are two main strategies used for group recommendation: (a) creating a group profile by combining the individual members' preferences into a single profile and (b) generating and then aggregating, into a single set of recommendations, lists for each individual member of the group [16, 3].

Merging Profiles. If we have two users, who have respectively rated $\{i_1, i_2\}$ and $\{i_3, i_4\}$, then their group profile is simply $\{i_1, i_2, i_3, i_4\}$. If, instead, there is an overlap in the group member's ratings, such as if they have rated $\{i_1, i_2\}$ and $\{i_1, i_3\}$, then the merged profile is $\{mean(i_1), i_2, i_3\}$. The resulting profile can then be directly used in any of the CF algorithms as if it were a single user.

Merging Recommendations. A variety of aggregation techniques have been proposed in the literature [18, 11, 16, 3]. We use the strategy known as *Least Misery* (LM). This approach seeks to minimise the probability that any one member of the group will strongly dislike the recommendations. More formally, once we have generated a set of recommendation lists for members of a group G, we set the relevance score of any item i as the smallest relevance score from amongst the group's individual's relevance scores. This means that the relevance of an item to a group is the least satisfied member's score.

Recent work has compared the two strategies: in [5], the authors found that the first approach marginally outperforms the second when using recipe ratings collected from a variety of families. In [3], instead, concludes that the LM strategy outperforms a range of other techniques. In this work, we use and test both strategies. All of the above approaches tend to assume that peoples' preferences do not change across being alone or being in a group. In this work, we will revisit this assumption by incorporating group membership into a probabilistic model: [2] also addressed this aspect of groups by explicitly integrating a *group disagreement* score into the groups' relevance rating score.

2.5 Evaluating Recommendations

Empirically measuring the quality of recommendations has, in the past, fallen into two camps. In the majority of cases, researchers have measured the accuracy of learning algorithms' predictions for hidden user ratings [17]. However, since those predictions will be used to rank items, ranking-based evaluations of CF have also been used [7]: in this work, we will focus on the latter, and our proposed model will be tailored towards ranking rather than predicting user ratings.

Evaluating the effectiveness of group recommendation algorithms has followed similar approaches to broader CF evaluations, but has suffered from the lack of available data on group preferences. Many studies have been conducted by synthetically creating different groups of various members based on user-similarities. For example, [3] studied the performance of different rank aggregation strategies by creating groups of different users based on similarities and generating recommendation lists for individual members and aggregating the lists. We overcome this by using a dataset that contains individual user preferences and their group membership. This dataset, from the German movie-rental site MoviePilot, was released as part of the Context-Aware Movie Recommendation Challenge at ACM RecSys 2011[3] and has been recently used to, for example, identify active members in a household [6].

3. INFORMATION MATCHING FOR GROUP RECOMMENDATION

As we noted in the previous section, a key aspect of group recommendation that is missing is the notion of matching items to users *as a group*, rather than as a *set of individuals*. In this section, we introduce a probabilistic model that seeks to address this problem. We describe the model via two steps: first (Section 3.1), we summarise the concept of information matching, and how it can be used for personalised recommendations to individuals. Then (Section 3.2), we augment the information matching model to include group relevance. In the following section, we then proceed to evaluate the effectiveness of this model against all of the algorithms that have been described in Section 2.

3.1 Information Matching

Much like traditional CF, the information matching model reasons upon *users* and *items* [13]. However, it does not represent these entities as sparse vectors of the numerical ratings they have input or received. Instead, each user and item is described with a set of binary features. More formally, we define a vector $E = \{\alpha_1, \alpha_2, \cdots, \alpha_f\}$ of user features, and a vector $F = \{\beta_1, \beta_2, \cdots, \beta_e\}$ of item features. A user is represented as $u \in \{0, 1\}^f$, or an f-dimensional binary feature vector over E, where $\alpha_k = 1$ if u is described by the k^{th} feature in E. Similarly, each item i is represented with an e-dimensional binary vector over F. As before, each $\beta_l = 1$ if i is described using l^{th} feature in F. The model makes no initial assumptions about what kinds of features exist: in practice, E and F may differ. In our case, individual users are described by item-features (i.e., E is the item space), and we treat users as features for the items (F is the set of users).

We assume that, as implicitly defined within any domain of recommendation, there exist *directional* relevance relationships between the users and items. User features have

[3]http://2011.camrachallenge.com/call-for-papers/

a *preference* toward item features: for example, a user described with a "child" feature may prefer items that have the "cartoon" feature. Similarly, item features have an *appeal* towards particular user features: as before, an item described with the "cartoon" feature may appeal to those users with the "child" feature. We formally define the binary preference matrix as:

$$M = \begin{array}{c} \\ \alpha_1 \\ \vdots \\ \alpha_f \end{array} \begin{array}{c} \begin{array}{cccc} \beta_1 & \beta_2 & \cdots & \beta_e \end{array} \\ \begin{pmatrix} 1/0 & 1/0 & \cdots & 1/0 \\ \vdots & \vdots & \ddots & \vdots \\ 1/0 & 1/0 & \cdots & 1/0 \end{pmatrix} \end{array}$$

Similarly, the binary appeal matrix is defined as:

$$N = \begin{array}{c} \\ \beta_1 \\ \vdots \\ \beta_e \end{array} \begin{array}{c} \begin{array}{cccc} \alpha_1 & \alpha_2 & \cdots & \alpha_l \end{array} \\ \begin{pmatrix} 1/0 & 1/0 & \cdots & 1/0 \\ \vdots & \vdots & \ddots & \vdots \\ 1/0 & 1/0 & \cdots & 1/0 \end{pmatrix} \end{array}$$

We note that M does not necessarily equal N^T. For example, a user described with feature "student" may have a preference for a car described with the "luxury item" feature, but the car described with the "luxury item" may not have appeal to users with the "student" feature[4].

In practice, the model assumes that varying degrees of uncertainty exist on features that describe u and i. Thus, we say that there is a probability that a user is described with a feature α_l, and represent this with $P(\alpha_l = 1|u)$. In our case, we assume that an individual's rating for an item is a stochastic function that combines the user-item preference with the item-user appeal. In other words, given a user u's rating r for item l, the feature α_l is:

$$P(\alpha_l = 1|u) \equiv P(\alpha_l = 1|r_l) \quad (4)$$

We assume that every individual user (and item) feature has a given Poisson rating distribution over the items (ref. users) that it describes, and another Poisson rating distribution on the items it does not describe. We opt for the Poisson distribution since it has successfully been used for term frequencies in information retrieval ranking [27] and since ratings tend to be small integers. We further assume that this user's ratings are the sole result of this feature description of items. Similarly, to estimate the item feature distribution over the kind of users, we assume that the observed ratings on this item are the result *only* of this item description of the user. These assumptions are clearly oversimplifications; we leave more sophisticated models to future work. Using these conditions, we compute the probability that the feature α_l describes user u as:

$$P(\alpha_l = 1|r) = \frac{P(r|\alpha_l = 1)P(\alpha_l = 1)}{\sum_{\alpha_l \in \{0,1\}} P(r|\alpha_l)P(\alpha_l)} \quad (5)$$

where r denotes user u's rating for the l^{th} item. We similarly compute $P(\beta_m = 1|i) \equiv P(\beta_m = 1|r)$. The Poisson distributions have parameters λ_1, λ_0, and a mixture probability p_1; we estimate these mixture parameters using the Expectation Maximisation algorithm [9]. In learning the mixture parameters values, we assign a "0" rating to all the items that the user never rated, which implies that the items were not relevant. While this may not be the case, this is common practice in CF when rating prediction is not involved [7]. By substituting the estimated mixture parameter values, we can compute:

$$\begin{aligned} P(\alpha_l = 1|u) &\equiv P(\alpha_l = 1|r) \\ &= p_{l_1} \frac{e^{-\lambda_{\alpha_{l_1}}} \lambda^r_{\alpha_{l_1}}}{\left(p_1 e^{-\lambda_{\alpha_{l_1}}} \lambda^r_{\alpha_{l_1}} + (1-p_1) e^{-\lambda_{\alpha_{l_0}}} \lambda^r_{\alpha_{l_0}}\right)} \end{aligned} \quad (6)$$

where $p_{l_1} = P(\alpha_l = 1)$, $\lambda_{\alpha_{l_1}}$ is the average value of α_l if l describes u, and $\lambda_{\alpha_{l_0}}$ is the average value if the feature α_l does *not* describe u. We similarly compute the probability for $P(e_m = 1|i)$. Finally, we calculate the relevance probability of a user-item pair (u, i), or $P(R = 1|u, i)$ as derived in [13]:

$$P(R = 1|u, i) \propto_R \prod_{<\alpha_l, \beta_m>} \underbrace{\frac{P(\alpha_l = 1|u)}{P(\alpha_l = 1)}}_{\text{part 1}} \underbrace{\frac{P(\beta_m = 1|i)}{P(\beta_m = 1)}}_{\text{part 2}} \quad (7)$$

Which captures the user description of feature l (Part 1) and the item description for feature m (Part 2). Once we compute the relevance score for each item for a given user using Equation 7, we rank all the items based on their probability of relevance.

In summary, information matching offers a different means of computing the relevance between a user and an item, by reasoning on the preference that the user may have for the item, and the appeal that the item may have for that user. In the following section, we build on this model so that it can further include the notion of groups of users.

3.2 Probabilistic Group Recommendation

We now describe how the model above can be augmented in order to provide group recommendations. We first define a group, describe different aspects of group recommendation, our notation, and provide a definition for item relevance to a group. Then, we describe a framework that uses information matching, sets of users, and group ranking strategies to compute item relevance to groups.

A *group* is a set of individuals who are set in a shared context (e.g., members of the same household, friends, etc.) and who may have common interests. Each individual will have both *personal* and *group* preferences: interests that they are likely to pursue as individuals, which may be unique and independent from others, and preferences that arise from being a member of the group, which will thus be shared with other group members. The relevance of an item to any given group will vary with the type of group that seeks recommendations. There are many different types of such groups that could be defined. For example:

1. **Consensus Preference Group**. If a group is a set of friends who want to do something together (e.g., go to the cinema), then the recommended items should be relevant to all individuals, or, at least, not disliked by any one member in those instances where no consensus is available.

2. **Shared Preference Group**. For groups that, again, seek items to be enjoyed together, but with conditions that are more relaxed than consensus preference

[4]This example assumes that, in general, students are not likely to be able to afford luxury items, and the car manufacturer is probably not targeting his product to students.

groups: recommended items should be relevant to all members, or at least not disliked by the majority when no consensus is available.

3. **Split Preference Group**. If a group consists of a household who wants recommendations as a unit, but may "consume" the items at different times (e.g., members of a family watching different television programs), then the relevance of an item will be determined by whether there exists at least one user who likes the item.

In the following, we use these notions of group relevance to extend the information matching probabilistic framework so that it caters for recommending items to groups. We define the following hypothesis: *The relevance between a group and an item i is only dependent on the relevance of i to individual members of the group.* Using this hypothesis, we derive a probabilistic group recommendation framework that not only includes the preferences of individual users but also integrates the users preferences when they are in a group while recommending a set of items.

First, we define the following notation:

1. G is a group containing a set of h users; $\Im$ is the set of users $\Im = \{u_1, u_2 \cdots, u_h\}$.

2. R_g is the binary *group* relevance between any group-item pair. R_g is 1 if the item is relevant to the group, and 0 otherwise.

3. $\Re$ is a vector of values containing the relevance of each user $u_j \in \Im$ to a given item i.

In order to recommend a set of items to a group G, we therefore need to calculate $P(R_g = 1|G, i)$ for each item, or the probability of relevance between the group G and each individual item i in collection. We derive this probability as follows. First, the probability of an item's relevance to a group is defined as:

$$P(R_g = 1|G, i) = \sum_{\Re} P(R_g = 1, \Im, \Re|G, i) \tag{8}$$

The Bayesian transformation allows to rewrite this as:

$$P(R_g = 1|G, i) = \sum_{\Re} \underbrace{P(R_g = 1|\Im, \Re, i, G)}_{\text{Part 1}} \underbrace{P(\Im, \Re|G, i)}_{\text{Part 2}} \tag{9}$$

We know that the R_g is dependent only on $\Im, \Re, i$, so we can ignore G in part 1, since group relevance of an item is dependent only on the relevance of each individual member of the group to that item (as per our hypothesis). Similarly, we can ignore G in part 2 as individual user's relevance given an item is independent of the group/groups s/he belongs to:

$$P(R_g = 1|G, i) = P(R_g = 1) \\ \sum_{\Re} P(\Re, u_1, \cdots, u_h, i|R_g = 1) P(\Re|u_1, \cdots, u_h, i) \tag{10}$$

Assuming that the relevance $R_j \in \Re$ is dependent only on u_j, i and eliminating the constant $P(R_g = 1)$, we can rewrite the above equation as:

$$P(R_g = 1|G, i) \propto_{R_g} \\ \sum_{\Re} \underbrace{\left\{ \prod_{j=1}^{h} P(R_j, u_j, i|R_g = 1) \right\}}_{\text{Part 1}} \underbrace{\left\{ \prod_{j=1}^{h} P(R_j|u_j, i) \right\}}_{\text{Part 2}} \tag{11}$$

Using Equation 11, we obtain a probabilistic model for group recommendation that accounts for both group (Part 1) and user (Part 2) relevance. The *Group* part of the Equation 11 captures the i relevance to u_j when she/he is in a group where as the *Individual* captures the relevance of i to u_j as an individual.

Assuming that the group recommendation scenario is to find items that all members like (i.e., the context is a consensus preference groups), we know that for $R_g = 1$ then $R_j = 1$ for each (u_j, i) where $u_j \in G$. By turning this implication around, we set $P(R_j = 1, u_j, i|R_g = 1) = 1$ and $P(R_j = 0, u_j, i|R_g = 1) = 0$. Finally, by substituting these into Equation 11 we get:

$$P(R_g = 1|G, i) \propto_{R_g} \prod_{j=1}^{h} P(R_j = 1|u_j, i) \tag{12}$$

We note that, by defining the problem space probabilistically, any mechanism to estimate the relevance between a user-item can be substituted into the framework. For example, if the context contains split preference groups, items could be ranked with:

$$P(R_g = 1|G, i) \propto_{R_g} \\ \min\{P(R_1 = 1|u_1, i), \cdots, P(R_h = 1|u_h, i)\} \tag{13}$$

Which captures the concept of *Least Misery* between the group members, and is the approach that we adopt in the following evaluation.

A crucial aspect of the computation of relevance in Equation 13 is that the probabilities of relevance between all the users in the group to all the items in the collection must be comparable, i.e. the probability space in which the probability of relevance between all user-item pairs is computed must be the same [24, 23]. This is uniquely achievable by using the principles of information matching [13]: most of the traditional relevance ranking functions used in information retrieval are not capable of estimating this relevance because they function on probability event spaces that are either conditioned on users (queries) [27] or items (documents) [30].

In summary, we outline the group recommendation algorithm that we have proposed here with the pseudocode in Algorithms 1 and 2. Algorithm 1 focuses on computing the user and item description vectors, and facilitates parallelising the algorithm for large-scale data. Algorithm 2, instead, computes the relevance of items to groups based on the least misery relevance of items to each member.

4. EVALUATION

In this section, we evaluate how the information matching model for group recommendation, introduced in the previous section, compares to the standard set of CF algorithms from Section 2: popularity, neighbourhood, and latent factor models.

Dataset	Users	Movies	Ratings	Rating scale	Label
MovieLens (100K)	983	1682	100,000	[1-5]	ML100K
MovieLens (1 Million)	6,040	3,952	1,000,000	[1-5]	ML1m
MoviePilot (Training)	171,670	23,974	4,391,822	[0-100]	MPtr
MoviePilot (Evaluation)	594	811	4,482	[0-100]	MPEval

Table 1: Dataset Descriptions. The number of users, items, and ratings as well as the two different rating scales used in each dataset. The "label" column denotes how the algorithms are referenced in the text and other tables.

Algorithm 1 Separately Calculate User and Item Vectors

```
Compute each user description vector
for $u \in Users\ U$ do
    for $l = 1 \cdots f$ do
        $E_u[l] = \frac{P(\alpha_l=1|u)}{P(\alpha_l=1)}$
    end for
end for
Compute each item description vector
for $i \in Items\ I$ do
    for $k = 1 \cdots e$ do
        $F_i[k] = \frac{P(\beta_k=1|i)}{P(\beta_k=1)}$
    end for
end for
```

Algorithm 2 Given group G, Calculate the relevance $\hat{r}$ for each item for the group by Least Misery

```
$t$ = relevance threshold
$X = (u,i) \in\ User \times Items$ where $r(u,i) \geq t$
for $u \in G$ do
    for unrated $i \in Items\ I$ do
        $\hat{r}(u,i) = 1$
        $\hat{r}(G,i) = \infty$
        for $(m,n) \in X$ do
            $\hat{r}(u,i) = \hat{r}(u,i) \times (E_u[l] \times F_i[k])$
        end for
        if $\hat{r}(G,i) \geq \hat{r}(u,i)$ then
            $\hat{r}(G,i) = \hat{r}(u,i)$
        end if
    end for
end for
```

Section 4.1 introduces the datasets that we used: a MoviePilot dataset of households' ratings for movies for group recommendation, and the widely used MovieLens datasets, that allow us to provide results that are comparable with the literature. Section 4.2 describes the methodology and metrics that we use; finally, Section 4.3 details all the results we obtained.

4.1 Datasets: MoviePilot and MovieLens

We use three different movie-rating datasets: the two publicly available MovieLens datasets (which differ in size) [20], and a MoviePilot dataset that was released as part of the Context-Aware Movie Recommendation 2011 Challenge at ACM RecSys [1], or CAMRa. We note that the main task of the first track of the challenge was to address the same problem[5] that we describe here; that is, recommending a given (*mp*) set of items to a household of users.

[5]http://2011.camrachallenge.com/call-for-papers/

The characteristics of the datasets are given in Table 1: there is a wide range in both the number of users and movies, and the MoviePilot data uses a 0 to 100 rating scale (as opposed to MovieLens' 1-5 star ratings). Furthermore, the MoviePilot data contains 290 unique households with between two to four members: majority of the user-ids have been assigned to a household. We note that, although previous research has used the MovieLens data to examine group recommendation scenarios, these datasets do not include any explicit group membership data. In the past, researchers have overcome this by forming implicit groups in the data [3]. However, since the MoviePilot data does contain group membership features, we solely use the MovieLens data to provide results for recommendations to individuals that allows our model to be compared with the CF literature.

4.2 Methodology and Metrics

Given the datasets above, we now describe how we tested and measured the efficacy of the recommendation algorithms described in Sections 2 and 3. We tested the algorithms in two settings: how well they can produce recommendations for individuals, and how well they produce recommendations for groups. We then compare these two settings to examine how much performance is lost between the two scenarios.

Recommendations to Individuals. First, we verified how our model performs when used to compute recommendations for *individuals*, compared to the other approaches. The main purpose for doing so is two-fold: (a) in previous sections we posited that better group recommendations could be obtained by more accurate models of group members, and (b) we demonstrate that we compare against strong baselines by reproducing results from [4].

In our evaluation, we rank *all* the items that each target user has not rated in the training set: for ML100K, ML1M and MoviePilot we ranked more than 1500, 3800, 23000 items respectively. We then set those test set items that the user rated higher than a given threshold (e.g., 4 stars or higher on the 1-5 scale) as relevant and all other items, both those rated below the threshold or not rated at all, as not relevant. We note that, in doing so, we will tend to observe very low performance scores. We rank all the items that were not rated by the user for two reasons: (1) it is more closer to the task that practical systems must do, and (2) it produced results with a more accurate distinction between different models, in terms of performance.

To conduct our experiments with the MovieLens data, we have randomly divided the datasets into training (60% of the data) and test sets, making sure that ratings from any given user are in both training and testing. We repeat this process five times to compute 5-fold cross validated results. We also used the MoviePilot data, by disregarding the group memberships.

Individual Recommendation Metrics. We evaluate performance with the following ranking metrics. First, for every user we compute the precision at rank position N defined as:

$$P@N = \frac{rel_N}{N} \quad (14)$$

where N is the length of the ranked list and rel_N is the number of items with user rating greater than the relevance threshold rating in the ranked list. The precision metric evaluates how well a model performs in putting relevant items in a top-N recommendation list, regardless of its rank.

Second, we measure the Normalised Discounted Cumulative Gain (nDCG). This metric measures the goodness of ranked list by considering the ratings for the lists' items with the discounted cumulative gain (DCG). We denote the rating that user u gives item i as $r(u, i)$. The discounted cumulative gain for user u at rank N, or DCG_N^u, is computed as:

$$DCG_N^u = r(u, i_1) + \sum_{j=2}^{N} \frac{r(u, i_j)}{log_2(j)} \quad (15)$$

We denote $IDCG_N^u$ as the maximum gain value for the user that is obtained with the optimal re-ordering of the N items. We use this value to produce the normalised discounted cumulative gain, $nDCG_N^u$:

$$nDCG_N^u = \frac{DCG_N^u}{IDCG_N^u} \quad (16)$$

In computing the nDCG score for each user, we assign a "0" rating to all the items that were not rated by the user, so that if the model recommends any item that was not rated by the user it will be penalised. To evaluate the global performance, we compute the average value of each metric over all the users. We denote the overall precision and nDCG at N with $P@N$, $NDCG@N$.

Finally, we also compute the system oriented performance metric Mean Average Precision [14], or MAP, which is defined as:

$$MAP = \frac{1}{|U|} \sum_{j=1}^{|U|} \frac{1}{|L_j|} \sum_{k=1}^{|L_j|} Precision(RL_{jk}) \quad (17)$$

where U is the set of users, L_j is the set of relevant items for the user j and RL_{jk} is the ranked list of items until k^{th} relevant item for the user j. $Precision(RL_{jk})$ for user j is computed using $precision@|RL_{jk}|$.

Recommendations to Groups. Next, we experiment with the extent that the algorithms can produce quality recommendations for *groups*, using the MoviePilot data. The methodology that we adopted sought to align itself to the structure of the CAMRa challenge. The MoviePilot data contains a set of households H came already divided into training and evaluation sets: the task of the challenge was to recommend a specified number of items to each household; data splitting and the specification of number of items to recommend to each household was defined by the organisers; the range of number of items recommended to each household is between 1 and 82. We thus use the training set (MPtr) to learn the model and rank all the non-rated items for each user and compare to the evaluation set (MPEval) for both the recommendation tasks.

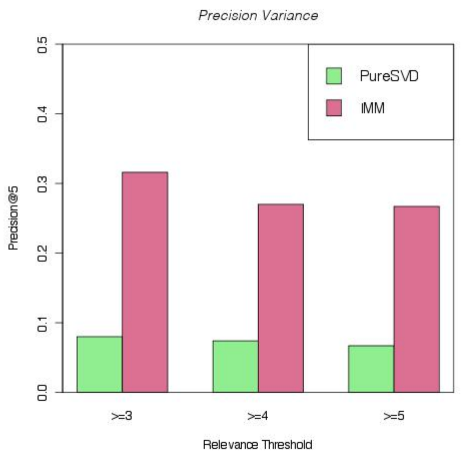


Figure 1: Recommending to Individuals. Precision@5 variance in performance, when using PureSVD and the IMM on the MovieLens 100k dataset, as the threshold of relevance is changed.

Group Recommendation Metrics. To evaluate the quality of group recommendation we use group precision at rank position mp, grP@mp, for the group recommendation list and the Mean Average Precision (MAP) for the group, where mp is the number of items recommended to each household as defined in the CAMRa data set. Group MAP is computed as:

$$gMAP = \frac{1}{|H|} \sum_{j=1}^{|H|} \frac{1}{|L_j|} \sum_{k=1}^{|L_j|} Precision(RL_{jk}) \quad (18)$$

where H is a set of households, L_j is the set of relevant set of items for the household j and RL_{jk} is the ranked list of items until the k^{th} relevant item for H_j. $Precision(RL_{jk})$ is computed as the average of the precisions of individual members of the household H_j.[6]

4.3 Experimental Results

Since we will be providing both reference (individual recommendation) and group recommendation results, in the following we describe each evaluation strategy separately. In the final subsection, we compare these two sets of experiments to examine how group and individual recommendation contexts differ.

4.3.1 Recommendations for Individuals

Table 2 summarises the performance of each model on different collections, with respect to each evaluation metric, when an item is considered to be relevant if the rating given by the user is 5 for MovieLens and greater than 70 for the MoviePilot data. Most notably, we have only reported MAP scores for the MoviePilot data. The reason for this is that the performance of the neighbourhood and latent factor models was close to 0[7]. Furthermore, the simple popularity based model (*Pop-item*) often produces better results than

[6]http://2011.camrachallenge.com/evaluation/

[7]A similar result was reported on the competition web page: `http://2011.camrachallenge.com/2011/11/final-evaluation/`

		Algorithm				
Collection	Metric	k-Item	k-User	PureSVD	*Pop-item*	IMM
ML100K	P@5	0.00135	0.006	0.067	0.227	0.267
	NDCG@5	0.0036	0.0091	0.0566	0.216	0.245
	MAP	0.013	0.041	0.061	0.119	0.156
ML1m	P@5	0.00047	0.071	0.093	0.149	0.175
	NDCG@5	0.0009	0.094	0.113	0.233	0.234
	MAP	0.00677	0.0320	0.063	0.109	0.128
MPEval	MAP	0.00061	0.00238	0.004	0.12	0.208

Table 2: Recommending to Individuals. Performance results, in terms of Precision@5, Non-Discounted Cumulative Gain@5 and MAP, when using the popularity baseline, neighborhood approaches (item and user based), PureSVD, and the Information Matching Model (IMM) in order to generate recommendations for individual users.

the personalised alternatives. However, we clearly observe that the IMM performs better than the all the remaining models on all the metrics. We also note that the IMM performance on P@10, P@20, NDCG@10, NDCG@20 is also higher when compared with the other models.

We have also tested the extent that performance on individual recommendation varies when we change relevance threshold from 3 to 5 (MovieLens) or 70 to 100 (MoviePilot). Figure 1 shows the performance of both PureSVD and IMM across these thresholds: the precision increases as the relevance threshold rating is reduced, but the IMM continues to show better performance than the PureSVD approach.

In conclusion, the individual recommendation results show that the two best performing personalised recommender models are the PureSVD and IMM. In the following section, we show how similar results emerge in the group recommendation scenario.

4.3.2 Recommendations for Groups

We evaluate group recommendation with our models when using both of the previously described strategies: (a) by creating profile for each group based on merging all members' ratings, and (b) by merging members' ranked list of items using *Least Misery* strategy. As before, we have also repeated the experiments for various relevance rating thresholds.

Table 3 summarises the performance of each model; we exclude the results from both the kNN user- and item-based approaches as they were all zeroes. Once again, it is clear that the group recommendation model based on the IMM outperforms the other two methods.

We note that the MoviePilot data does not contain the group information for all the users in the training data. So, when we merge the group profiles the items considered in training were the items rated by at-least one member who has a group identifier. As a result, we lack the rating information of the items that were not rated by any member of any group, and the total number of items ranked for each user is less than the total number potential test itemscompared to the other two methods. From the Table 3, we can observe that the performance of the profile aggregation model is higher than the performance obtained for the merging ranked lists model. However, the group grP@mp score is higher for the merging ranked lists model than the aggregated user profile.

4.3.3 Performance Loss Between Individual and Group Recommendation

Finally, we measured how much recommendation quality degrades between the individual and group scenarios. Figure 2 puts the relative performance between individual and group recommendation of the PureSVD and IMM (in terms of P@5), as different relevance rating thresholds were tested.

It is clear that, throughout all approaches, a loss is incurred when transitioning from individual to group recommendations. This captures the difficulty of the group recommendation problem, and may be explained by the number of different assumptions that each algorithm must make. However, not only does the IMM obtain better results compared to the PureSVD: it also degrades less when used for group recommendation. In fact, precision wanes by up to 26% when IMM is used for group recommendation; the PureSVD approach, instead, loses between $50 - 95\%$ of the precision it had for individuals when applied to groups.

5. DISCUSSION

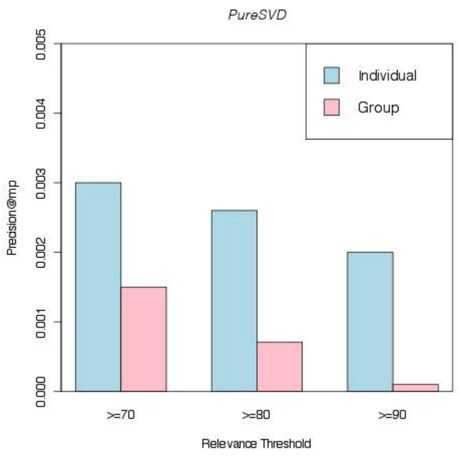
In the previous section, we observed how the IMM outperforms the latent factor approach, when computing ranked recommendations for both individuals and groups. In this section, we further discuss the differences between these algorithms, and how these may affect their performance.

The IMM differentiates itself from latent factor and neighbourhood based collaborative filtering in a number of ways. First, the model does not require a common feature space. It allows for the users and items to be modelled independently of one another (if necessary), and can incorporate any available information that is solely specific to users or items. Unlike dimensionality reduction methods, such as matrix factorisation, Latent Factor Models [17] and topic models [15], we do not need to set any specific number of hidden dimensions in which both the users and items will be represented. In other words, it does not involve a lower dimensional representation of features.

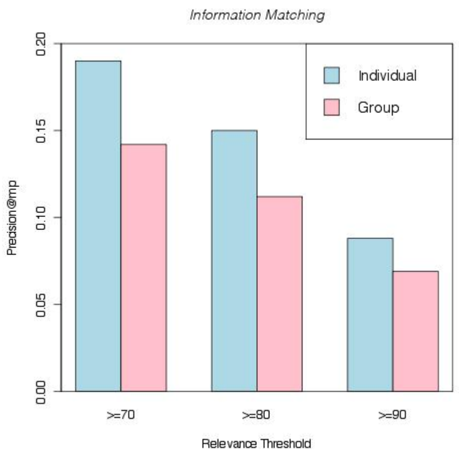
This approach to relevance seamlessly captures similarities between users and between items. For a given user-item pair, the formulation incorporates information about other users who have the same features (i.e. items) as this user, and other items which have the same features (i.e. users) as this item. This means that the ranking function implicitly uses information from similar users to this one and from similar items to the target, through a set of relevant user-item pairs. So, there is no need to explicitly compute the

Model	Metric	Merging Individuals' Profiles > 70	>80	>90
PureSVD	grP@mp	0.00027	0.00013	0.000034
	gMAP	0.00416	0.00345	0.00221
		Rank Aggregation By Least Misery		
	Metric	> 70	>80	>90
PureSVD	grP@mp	0.0015	0.00071	0.0001
	gMAP	0.0029	0.0021	0.0017
		IMM based Least Misery		
	Metric	> 70	>80	>90
IMM	grP@mp	0.142	0.112	0.068
	gMAP	0.159	0.147	0.120

Table 3: Group Recommendation. Performance results on the MoviePilot data for varied relevance rating thresholds. We include group precision (grP@mp) and Group Mean Average Precision (gMAP) for (a) PureSVD on group profiles that are merged ratings of group members, (b) individual PureSVD results that were merged with *Least Misery*, and (c) the group information matching model.



(a) PureSVD



(b) IMM

Figure 2: Performance Loss Between Individual and Group Recommendations. We plot, for PureSVD and IMM, the performance change as the relevance threshold is varied. Note that the rating scales for each plot are different, and that the PureSVD consistently has a higher loss in performance when used for group recommendation (compared to when it is used for individual recommendation).

similarities between the users or items, which is the basis of the user-based approaches [26] and the item-based approaches [25]. The model is also capable of incorporating other kinds of features; for example, features of the users other than the items they like. In principle, the preference and appeal matrices can map between any sets of features; we did not pursue this aspect in this work due to the lack of data. Finally, we note that the two components of the ranking function 7 (parts 1 and 2 in Equation 4) can be treated independently of each other. This also facilitates parallelising the algorithm to run on large scale data sets.

6. CONCLUSION

Web recommender systems are increasingly being applied to group recommendation scenarios, both in instances where users may share online accounts or may seek recommendations for shared activities. Recent research in this domain, however, has addressed this scenario by computing recommendations for individuals and later merging those recommendations together; in doing so, these approaches exclude the effect of group dynamics on users' preferences. Therefore, we have introduced a probabilistic relevance framework for group recommendation that can integrate any group recommendation strategy while taking into account the preferences of the group as a whole. The model can also include any group-specific features (e.g., group-type, such as *family* or *friends*) that may not be captured in individual's profiles: in future work, we would like to investigate this further. In particular, future work should seek to understand what additional features matter for group recommendation, and to discover what contextual features may influence a group's preferences.

We evaluated our approach, particularly focused on the *Least Misery* strategy, using available data that contains the ratings that users (who belong to given households) have given to movies. Our future work includes extending this evaluation to examine other techniques as well. We found that our information-matching approach outperformed a popularity based baseline, neighbourhood models, and a latent factor model across a range of ranking metrics when recommending to groups: we therefore also tested the model on the MovieLens dataset for *individual* recommendations in order to ensure that our results are aligned with others reported in the literature.

7. REFERENCES

[1] *CAMRa '11: Proceedings of the 2nd Challenge on Context-Aware Movie Recommendation*, New York, NY, USA, 2011. ACM.

[2] S. Amer-Yahia, S. B. Roy, A. Chawlat, G. Das, and C. Yu. Group recommendation: semantics and efficiency. *Proc. VLDB Endow.*, 2(1):754–765, Aug. 2009.

[3] L. Baltrunas, T. Makcinskas, and F. Ricci. Group recommendations with rank aggregation and collaborative filtering. In *Proceedings of the fourth ACM conference on Recommender systems*, RecSys '10, pages 119–126, New York, NY, USA, 2010. ACM.

[4] A. Bellogín, P. Castells, and I. Cantador. Precision-oriented evaluation of recommender systems: an algorithmic comparison. In *RecSys*, 2011.

[5] S. Berkovsky and J. Freyne. Group-based recipe recommendations: analysis of data aggregation strategies. In *Proceedings of the fourth ACM conference on Recommender systems*, RecSys '10, pages 111–118, New York, NY, USA, 2010. ACM.

[6] P. G. Campos, A. Bellogin, F. Díez, and I. Cantador. Time feature selection for identifying active household members. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, CIKM '12, pages 2311–2314, New York, NY, USA, 2012. ACM.

[7] P. Cremonesi, Y. Koren, and R. Turrin. Performance of recommender algorithms on top-n recommendation tasks. In *RecSys*, RecSys '10, 2010.

[8] A. Crossen, J. Budzik, and K. J. Hammond. Flytrap: intelligent group music recommendation. In *Proceedings of the 7th international conference on Intelligent user interfaces*, IUI '02, pages 184–185, New York, NY, USA, 2002. ACM.

[9] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the em algorithm. *J. Royal Statistical Society, Series B*, 1977.

[10] M. Deshpande and G. Karypis. Item-based top-n recommendation algorithms. *ACM Trans. Inf. Syst.*, 22(1):143–177, Jan. 2004.

[11] C. Dwork, R. Kumar, M. Naor, and D. Sivakumar. Rank aggregation methods for the web. In *Proceedings of the 10th international conference on World Wide Web*, WWW '01, pages 613–622, New York, NY, USA, 2001. ACM.

[12] D. Goldberg, D. Nichols, B. M. Oki, and D. Terry. Using collaborative filtering to weave an information tapestry. *Commun. ACM*, 35(12):61–70, Dec. 1992.

[13] J. Gorla, S. Robertson, J. Wang, and T. Jambor. A theory of information matching. *CoRR*, abs/1205.5569, 2012.

[14] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Trans. Inf. Syst.*, 22(1):5–53, Jan. 2004.

[15] T. Hofmann. Latent semantic models for collaborative filtering. *ACM Trans. Inf. Syst.*, 2004.

[16] A. Jameson and B. Smyth. The adaptive web. chapter Recommendation to groups, pages 596–627. Springer-Verlag, Berlin, Heidelberg, 2007.

[17] Y. Koren, R. M. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *IEEE Computer*, 2009.

[18] 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, Feb. 2004.

[19] K. McCarthy, M. Salamó, L. Coyle, L. McGinty, B. Smyth, and P. Nixon. Cats: A synchronous approach to collaborative group recommendation. In *FLAIRS Conference*, pages 86–91, 2006.

[20] B. N. Miller, I. Albert, S. K. Lam, J. A. Konstan, and J. Riedl. Movielens unplugged: experiences with an occasionally connected recommender system. In *Proceedings of the 8th international conference on Intelligent user interfaces*, IUI '03, pages 263–266, New York, NY, USA, 2003. ACM.

[21] M. O'Connor, D. Cosley, J. A. Konstan, and J. Riedl. Polylens: a recommender system for groups of users. In *Proceedings of the seventh conference on European Conference on Computer Supported Cooperative Work*, ECSCW'01, pages 199–218, Norwell, MA, USA, 2001.

[22] S. Pizzutilo, B. De Carolis, G. Cozzolongo, and F. Ambruoso. Group modeling in a public space: methods, techniques, experiences. In *Proceedings of the 5th WSEAS International Conference on Applied Informatics and Communications*, AIC'05, pages 175–180. World Scientific and Engineering Academy and Society, 2005.

[23] S. Robertson. The unified model revisited. In *SIGIR 2003 Workshop on Mathematical/Formal Models in Information Retrieval*, 2003.

[24] S. Robertson. On event spaces and probabilistic models in information retrieval. *Inf. Retr.*, 8(2):319–329, Apr. 2005.

[25] B. Sarwar, G. Karypis, J. Konstan, and J. Reidl. Item-based collaborative filtering recommendation algorithms. In *WWW*, 2001.

[26] U. Shardanand and P. Maes. Social information filtering: algorithms for automating ẅord of mouth ̈. In *CHI'95*, 1995.

[27] R. Stephen and Z. Hugo. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends in Information Retrieval*, 3(4), 2009.

[28] X. Su and T. M. Khoshgoftaar. A survey of collaborative filtering techniques. *Adv. in Artif. Intell.*, 2009:4:2–4:2, Jan. 2009.

[29] 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, Mar. 2006.

[30] C. Zhai. Statistical language models for information retrieval: Critical review. *Foundations and Trends in Information Retrieval*, 2(3):137–213, 2008.