Pairwise Learning in Recommendation: Experiments with Community Recommendation on LinkedIn

Amit Sharma
Dept. of Computer Science
Cornell University
Ithaca, NY 14853 USA
asharma@cs.cornell.edu

Baoshi Yan
LinkedIn Corp.
Mountain View, CA 94043 USA
byan@linkedin.com

ABSTRACT

Many online systems present a list of recommendations and infer user interests implicitly from clicks or other contextual actions. For modeling user feedback in such settings, a common approach is to consider items acted upon to be relevant to the user, and irrelevant otherwise. However, clicking some but not others conveys an implicit ordering of the presented items. Pairwise learning, which leverages such implicit ordering between a pair of items, has been successful in areas such as search ranking [12, 14]. In this work, we study whether pairwise learning can improve community recommendation. We first present two novel pairwise models adapted from logistic regression. Both offline and online experiments in a large real-world setting show that incorporating pairwise learning improves the recommendation performance. However, the improvement is only slight. We find that users' preferences regarding the kinds of communities they like can differ greatly, which adversely affect the effectiveness of features derived from pairwise comparisons. We therefore propose a probabilistic latent semantic indexing model for pairwise learning (Pairwise PLSI), which assumes a set of users' latent preferences between pairs of items. Our experiments show favorable results for the Pairwise PLSI model and point to the potential of using pairwise learning for community recommendation.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Information filtering; H.5 [**Online Information Services**]: Web-based services

Keywords

pairwise learning, community recommendation

1. INTRODUCTION

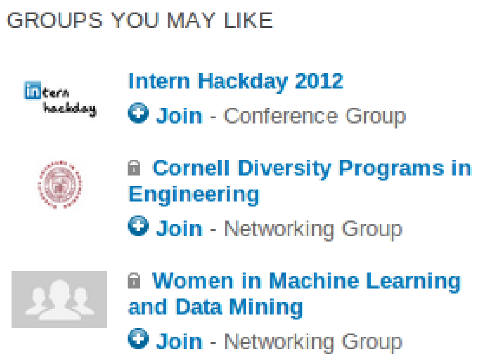
An increasing number of recommendation systems on the web depend on implicit binary feedback – the presence or absence of a user action on an item. Examples of binary user actions include liking an item, following a person, joining a community or clicking through a news story. In such systems, user feedback differs from traditional rating-based systems in two important ways–feedback is only partial (negative ratings are implicit), and always binary. The presence of only positive feedback creates a challenge in learning user preferences.

One intuitive approach is to consider items with positive feedback as relevant, and consider all the other items as irrelevant. This is a common approach in collaborative filtering, and is an effective way of understanding user feedback [4, 7]. However, collaborative filtering algorithms work best when given ratings across the good-bad spectrum, and even typical classification algorithms rely on a comparable number of positive and negative examples. To counter for the positive bias in feedback, another approach in literature has been to consider the problem as a special case of one-class classification and develop algorithms to generate artificial negative data [16, 15].

The above approaches, however, lose the relative nature of the information conveyed by the items that are shown to a user, but are not acted upon. Assuming a user sees all the recommendations and acts only on some of them, her inaction on certain items does not directly convey an absolute rejection of those items. Instead, we argue that it conveys an implicit negative feedback for those items, relative to the ones she acted upon. Thus, we can reasonably assume that clicked items are more relevant for the user than unclicked ones, though we cannot be sure about the absolute quality of unclicked items. This is the basis of pairwise learning, a popular approach for relevance feedback in search [3].

Past experiments with pairwise learning in collaborative recommendation systems have shown promising results [20, 17]. In this paper, we explore how a pairwise approach for preference learning can be used in a content-based recommender. In particular, we study the problem of recommending communities to users, where a community is a social unit consisting of people with certain common characteristics or interests, for example, a LinkedIn group for RecSys 2013, a LinkedIn group for Cornell alumni, or a Facebook page for RecSys. Our findings are three-fold. First, we present general pairwise adaptations for content-based recommendation based on logistic regression, and show that pairwise models are comparable and in general slightly better than the non-pairwise model. Second, we show that a direct pairwise adaptation might not yield significant improvement for community recommendation, due to users' vast different commu-

RecSys'13, October 12–16, 2013, Hong Kong, China.
Copyright 2013 ACM 978-1-4503-2409-0/13/10 ...$15.00.
http://dx.doi.org/10.1145/2507157.2507175.


GROUPS YOU MAY LIKE

Intern Hackday 2012
Join - Conference Group

Cornell Diversity Programs in Engineering
Join - Networking Group

Women in Machine Learning and Data Mining
Join - Networking Group

Figure 1: Group recommendation interface on linkedin.com

nity preferences which make pairwise comparison less meaningful. Third, we present a probabilistic latent semantic indexing model for pairwise learning, which models latent user preferences and significantly outperforms the direct pairwise adaptation models and the non-pairwise model.

Application domain. We choose community recommendation on LinkedIn.com as a domain for evaluating our recommendation models. LinkedIn is the world's largest professional network in which members create profiles and connect with other individuals. In addition, members are free to create and join communities (or *groups* [1]) of their liking (subject to admin approval, in certain cases). Members may add themselves to a group when they see it on another member's profile, through search or when they get recommendations. At present, there are more than 2 million groups on LinkedIn. The recommendation problem is to select a set of top-k groups to join for each user.

Figure 1 shows a typical recommendation interface. A user is presented with a list of k recommendations. A user conveys positive feedback by either clicking the link corresponding to a group, and/or choosing to join the group. However, the interface does not allow negative feedback, which provides a motivation for using pairwise learning.

2. RELATED WORK

We first discuss pairwise learning, and its relevance for recommendation systems. In the second part of this section, we present previous work on community recommendation.

2.1 Pairwise learning

Understanding implicit feedback has been an active problem for search engines. Joachims [9] was the first to show how information retrieval can be improved by considering user feedback in pairs of items. He assumed that users are more likely to see items ranked higher in the list (position bias [6]). Thus, between a clicked item and an unclicked item ranked higher than that item, we can infer safely that the clicked item is more relevant than unclicked one. These clicked and unclicked items in search logs are clubbed together to form pairs, which convey a relative ordering between their constituent items. Encoding user actions into pairwise preferences converts the problem into a classification problem on pairs, instead of classifying relevance for individual items. This approach was shown to improve retrieval results, and is now a popular method of relevance feedback in search engines [12, 14].

In principle, one could construct triplets, or even arbitrary lists [3] to encode relative preference. However, in a recent paper, Radinsky and Ailon [18] demonstrate that triplets do not give any more information than pairs in terms of the statistical significance of preference. Further, in a user study, Joachims et al. [10] provided empirical support for the pairwise approach by showing that people are more consistent at making comparisons between items than evaluating on an absolute scale of relevance or quality. The use of pairs is also consistent with an axiom in decision theory, which states that the relative attractiveness of two choices does not depend upon other choices available (Independence of Irrelevant alternatives [19]).

Apart from using click logs, pairs can also be constructed in other ways to convey relative preference of users. For example, Balakrishnan and Chopra [1] have proposed an adaptive scheme in which users are explicitly asked for their relative preference between a pair of items. Thus, users give pairwise feedback to the underlying algorithm, which updates user parameters as it receives more responses. Instead of pairs of items, one may also ask users to order a set of items [11]. Though it may provide an accurate measure of a user's preference, explicitly asking users for their preference may not be feasible for large numbers of users or items, or desirable as a design strategy in certain cases.

To the extent that recommendations are shown as an ordered list, the problem of recommendation quality is similar to that of relevance in search engines. Just as relevance in search depends on the information need, quality in recommendation depends on the user preference and context. For matrix factorization and kNN models, recommendation performance on movies has been shown to improve by incorporating pairwise preferences from recommendation feedback logs [20]. A pairwise strategy may also be applied to explicit ratings, where pairs are generated by comparing Likert-scale ratings of a user [13, 17]. In the present work, we focus on problem domains where such rich ratings are not present, using content-based recommendation models. As in [9], we use system logs of clicked recommendations to construct pairs.

2.2 Community Recommendation

We now present a brief overview of previous work on community recommendation. In one of the first papers on community recommendation, Spertus et al. [21] used similarity measures to recommend communities in Orkut which are similar to a particular community. Chen et al. [4] extended the work on Orkut communities and compared two different algorithms, Association Rule Mining and Latent Dirichlet Allocation for the purposes of community recommendation. In [5], Chen et al. describe a latent model that conditions a user joining based on the hidden topics that a community represents. In another domain (Flickr), different memory-based and model-based approaches have been compared in terms of top-k recommendation, and tags were found to be helpful [22].

Owing to the rich profile data (education, career, skills, interests, publications etc.) available for each user on LinkedIn, we adopt a content-based approach to recommendation. This

[1]We use the term community recommendation to disambiguate against the more common use of the term group recommendation to mean recommendation *for* groups.

is not unlike many other online social networks, where users self-create an elaborate personal profile. For example, Facebook hosts a profile for each user which spans a user's interests. LinkedIn groups, especially, is an example of a recommendation domain where we have rich data about a user as well as the group. We build on previous work by adapting a similarity-based algorithm (logistic) and a latent model (PLSI) with pairwise feedback and comparing their performance to non-pairwise models.

3. MODEL PRELIMINARIES

For clarity, let us first formally define the problem. Given a user u, the recommendation task is to suggest communities that may be of interest, from a set of candidate communities Y. Suppose there are $|U| = m$ users and $|Y| = n$ communities. Each observed preference is in the form of a tuple (u, y), where u is a user, y is a community, and $u \in \{1, 2, 3...m\}$, $y \in \{1, 2, 3...n\}$. We assume that we are given a set of known (u, y) and the task is to predict a set $R(u)$ for each user which denotes recommendations for a user u.

For our problem, we specify another variable s which denotes the impression list, or the communities that are shown together to a user. For example, in Figure 1, the three communities presented will belong to the same impression list. Thus each observation is given as a (u, y, s).

Current system. There are three main components: feature generation, model learning, and computation of the results from the learned model. Although continual feature generation and free-text standardization over millions of members is a complex problem in itself, here we provide a brief overview. Features for members are largely profile-driven, such as skills, industry, past positions, education etc. Similarly, a group is represented by features such as group summary and category. In addition, a group's features are also constructed in aggregate from the features of its constituent members. Elements of these two feature sets are matched to each other, to determine similarity between a user and a group. In the limiting case, the number of matchings can be quadratic in the number of features. Thus, individual features are matched only when semantically relevant. For example, it may be useful to match the group description with "interests" feature for a user, but not with the "connection counts" feature. For every user and group, a dot product of matched user-group features $(u.y)$ gives us a similarity score. In addition, we introduce social features that reflect user-group interactions, for instance, whether and how many friends of a user have joined a group. The final similarity vector conveys the degree of similarity between a user and a group. All models described utilize the similarity feature vector, thus constructed.

We now describe the current model in production on LinkedIn (as of May 2012), starting from some basic definitions. Given a similarity vector for each pair of user and item, a simple recommendation model can be described as:

$$R(u) = \{y \in Y : Sim(f_u, f_y) > \epsilon\}$$

for some $\epsilon > 0$, where $Sim(f_u, f_g)$ is the similarity vector for user u and group y feature vectors. If the similarity function is just a dot product of the vectors, we get:

$$Sim(f_u, f_y) = \sum_{i=1}^{n} w_i * f_u^{(i)} * f_y^{(i)}$$

where w_i represents the relative importance of different similarity features. Using the sigmoid function for similarity and setting $\epsilon = 0.5$, we can represent the problem in the form of standard logistic regression.

$$R(u) = \{y \in Y : \frac{1}{1 + e^{-Sim(f_u, f_y)}} > 0.5\} \qquad (1)$$

Given this model, we precompute the similarity between a users and all items. For each user, the most similar items can be chosen for recommendation. This is the baseline model implementation for our experiments.

Pairwise approach. We now turn to the preliminaries for using pairwise approach for recommendation. We first discuss how pairs are created from group-join data.

The first step is to access the group-join log as well as the recommendation impressions log. To connect a group-join with a recommendation impression, we choose the latest impression for each user-item pair and select all group-joins which occured within 10 minutes of the impression (a recommendation is not shown after a user joins a group). This is a conservative assumption, considering that the join may still happen on account of the impression or otherwise. But for the purposes of this study, a hard stance of 10 minutes is taken. We now have a a set of ranked impression lists. Each element of the ranked list contains an item and an indicator of whether it led to a group-join or not. From these lists, we create item pairs by the following rule (see [9]):

$$\begin{aligned} rank(I_{y_2}) > rank(I_{y_1}) \ and \ join(y_2) \ and \ !join(y_1) \\ \Rightarrow pair(y_1, y_2) \end{aligned} \qquad (2)$$

where I_{y_2} and I_{y_1} represent impressions in the same list for $y2$ and $y1$. Once we have the pairs, we define a variable t for each user and a pair of items (y_1, y_2), such that the learning problem translates to a classification problem on t.

$$t_u(y_i, y_j) = \begin{cases} 1 & \text{if } y_j \text{ is preferred over } y_i \\ 0 & \text{otherwise} \end{cases}$$

given a user u and two items y_1 and y_2.

Note that our above strategy for creating pairs will lead to pairs where the first group in the pair is always higher in preference ($t = 1$) than the second group. To ensure equal number of positive and negative examples for classification, we randomly invert all pairs with probability 0.5.

Any classification algorithm can be used to learn preferences on pairs, thus constructed. However, an important requirement is to have a fast way of ranking individual items, once we have a learnt a model for classifying pairs. For a naive implementation, ranking k items will require constructing $O(k^2)$ pairs, which will be too computation intensive. For the best performance, the ranking function should be independent of the pairs. In the next two sections, we discuss how we can derive fast ranking functions for logistic regression and PLSI-based pairwise models.

4. LOGISTIC PAIRWISE MODELS

We now describe two pairwise models derived from logistic regression. We assume that we are given a list of M known pairwise preferences $t_u(y_i, y_j)$. Let us define an item ranking function $h_\theta(y)$, which has the following property (as in [9]):

$$h_\theta(y_i) > h_\theta(y_j) \Leftrightarrow y_i \ is \ preferred \ over \ y_j$$

Then our task is to learn such a function h_θ from the available pairwise preferences. We make a simplifying assumption that pairwise preferences are independent of each other, drawing on the Independence of Irrelevant Alternatives assumption from decision theory literature [19]. Thus, our task can be set-up as a likelihood maximization problem:

$$\textit{maximize } \Pi_{k=1}^{M} P(t|y_i, y_j, u; \theta)$$

If we redefine t in terms of h,

$$t_u(y_i, y_j) = \begin{cases} 1 & \text{if } h_\theta(y_j) \geq h_\theta(y_i) \\ 0 & \text{otherwise} \end{cases}$$

then we can write the equivalent maximization problem.

$$\textit{maximize } \Pi_{k=1}^{M} P[h_\theta(y_j) \geq h_\theta(y_i)]^t P[h_\theta(y_j) < h_\theta(y_i)]^{(1-t)}$$

$$\textit{maximize } \Pi_{k=1}^{M} P[h_\theta(y_j) \geq h_\theta(y_i)]^t (1 - P[h_\theta(y_j) \geq h_\theta(y_i)])^{(1-t)}$$

Based on how we model $h_\theta(y)$, the above formulation gives rise to two novel pairwise models.

4.1 Feature Difference Model

In the special case that $h_\theta(y)$ is a linear function, we have $h_\theta(y) = \theta^T y$. Then the problem reduces to:

$$\textit{max } \Pi_{k=1}^{M} P[\theta^T(y_j - y_i) \geq 0]^t (1 - P[\theta^T(y_j - y_i) \geq 0])^{(1-t)}$$

Taking $y_{ji} = y_j - y_i$, this formulation is the same as standard logistic regression with features y_{ji}. Hence if we assume that the ranking function is linear, we can simply learn a logistic model on the difference of features of the items.

To rank the items, we use the linearity property again. We can see that $h_\theta(y_j) \geq h_\theta(y_i) \Leftrightarrow y_j \geq y_i$. Thus, it is sufficient to calculate ${\theta^*}^T y_i$ for each item to rank the items.

4.2 Logistic Loss Model

The feature difference model lends itself well to a standard learning framework, but makes a strong assumption about the ranking function. We now propose a model for any general ranking function. We define:

$$g_\theta(y_i, y_j) = h_\theta(y_j) - h_\theta(y_i)$$

as the difference in ranking scores for two items y_i and y_j. Then, we can write $P(t|y_i, y_j)$ as (where t is as before):

$$P(t = 1|y_i, y_j; \theta) = \frac{g_\theta(y_i, y_j) + 1}{2}$$

Since $h_\theta(y) \in [0, 1]$, we add the additional constants to ensure that the $P(t|y_i, y_j) \in [0, 1]$. This formulation is based on the intuition that difference in ranking scores for two items is proportional to the probability of one being more preferred than the other.

We can now frame this as a maximum likelihood estimation problem.

$$\textit{max } \Pi_{k=1}^{M} (\frac{g_\theta(y_i, y_j) + 1}{2})^t (1 - \frac{g_\theta(y_i, y_j) + 1}{2})^{1-t}$$

$$l(\theta) = \textit{max} \sum_{k=1}^{M} t \log[g(y_i, y_j) + 1] + (1 - t) \log(1 - g(y_i, y_j))$$

Using gradient descent,

$$\frac{\partial l(\theta)}{\partial \theta_n} = \sum_i (\frac{2t - g(y_i, y_j) - 1}{1 - g(y_i, y_j)^2}) \frac{\partial g(y_i, y_j)}{\partial \theta_n}$$

We can use any ranking function h_θ in the above equation. For our experiments, we choose the logistic function because it is used as the base for LinkedIn recommendation platform. Setting $h_\theta(y) = \frac{1}{1+\exp(-\theta^T y)}$, we get the gradient in Equation 3 (Figure 2).

The likelihood can be optimized using any of the gradient methods. We use L-BFGS [2] in our implementation. For generating top recommendations, we have to order items by $h_{\theta*}(y)$. Since logistic is an increasing function, this reduces to ranking items based on their ${\theta^*}^T y$ scores, as in the Feature Difference model.

5. PAIRWISE PLSI MODEL

Given the success of pairwise learning in domains such as information retrieval, it is natural to expect similar improvement in recommendation. However, as that will be shown later in Section 6, the two pairwise models described above only slightly outperform a baseline non-pairwise model.

One of the major reasons for the limited improvement is the heterogeneity in communities and users' vast different preferences for the communities they want to join, which make features derived from pairwise comparison less effective. For example, a user might look to expand her network. Thus even though a relevant interest-based group (e.g, a machine learning group) is recommended, the user might choose an alumni group instead, which as a result of pairwise comparison would penalize the feature representing user interest and reward the feature representing user's education history. When presented with a similar list of recommendations, however, another user might choose an interest group over an alumni group, thus producing the opposite impact on the two features. In the extreme case that an equal number of users make the opposite choices, the two features representing interest and education would not serve as discriminative features as we would have expected.

We now present a pairwise model that also accounts for variation in users' preferences towards different features, by treating a user as a mixture of some latent preferences. Our model is based on probabilistic latent semantic indexing [8]. The intuition is that users may have different reasons for joining a group, such as networking, learning skills or for social support [3]. Each user may also value these reasons differently. Assuming the possible reasons people join groups are consistent across users, we can think of each user's decision to join a group as a combination of these reasons, or *core preferences.*

Figure 3 shows the plate notation for our proposed model. We assume that each user can be described by a set of Z core latent preferences. We also assume that given a core preference z and two items y_i and y_j, the preference value t for the two items is independent of the user. Thus, when presented with a pair of communities (y_1,y_2), a user's decision is computed from the mixture of core preferences, weighted by the relative importance of each core preference to the user.

[2]http://www.chokkan.org/software/liblbfgs/

[3]We verified this intuition by considering different types of groups on LinkedIn (networking, professional, alumni, philanthropy, corporate and conference) as proxies for reasons to join. Using a random sample of 1M users, we found that most users join groups of more than one type; mean number of group types per user is 3.1 and median is 3.

$$\frac{\partial l(\theta)}{\partial \theta_n} = \sum_i \frac{2t - g(y_i, y_j) - 1}{1 - g(y_i, y_j)^2}(h_\theta(y_j)(1 - h_\theta(y_j))y_{jn} - h_\theta(y_i)(1 - h_\theta(y_i))y_{in}) \tag{3}$$

$$\textbf{E Step: } Q_k(z) = \frac{(h_{\theta_z}(y_j) - h_{\theta_z}(y_i) + 1)^t(1 - (h_{\theta_z}(y_j) - h_{\theta_z}(y_i))^{1-t}\exp(w_z^T u^{(k)})}{\sum_{z'}(h_{\theta_{z'}}(y_j) - h_{\theta_{z'}}(y_i) + 1)^t(1 - (h_{\theta_{z'}}(y_j) - h_{\theta_{z'}}(y_i))^{1-t}\exp(w_{z'}^T u^{(k)})} \tag{4}$$

$$\textbf{M Step: } \begin{aligned} &maximize_{\theta,w} \sum_k \sum_z Q_k(z)[w_z^T u^{(k)} - \log(\sum_{z'} \exp(w_{z'}^T u^{(k)}))+ \sum_{k\{t=1\}} \sum_z Q_k(z) \log(\frac{h_{\theta_z}(y_j^{(k)}) - h_{\theta_z}(y_i^{(k)}) + 1}{2}) \\ &+ \sum_{k\{t=0\}} \sum_z Q_k(z) \log(\frac{1 - (h_{\theta_z}(y_j^{(k)}) - h_{\theta_z}(y_i^{(k)}))}{2}) \end{aligned} \tag{5}$$

$$\begin{aligned} \frac{\partial l(\theta, w)}{\partial \theta_{zk}} &= \sum_{k\{t=1\}} Q_k(z) \frac{h_{\theta_z}(y_j^{(k)})(1 - h_{\theta_z}(y_j^{(k)})y_{jk} - h_{\theta_z}(y_i^{(k)})(1 - h_{\theta_z}(y_i^{(k)})y_{ik}}{h_{\theta_z}(y_j^{(k)}) - h_{\theta_z}(y_i^{(k)}) + 1} \\ &- \sum_{k\{t=0\}} Q_k(z) \frac{h_{\theta_z}(y_j^{(k)})(1 - h_{\theta_z}(y_j^{(k)})y_{jk} + h_{\theta_z}(y_i^{(k)})(1 - h_{\theta_z}(y_i^{(k)})y_{ik}}{1 - (h_{\theta_z}(y_j^{(k)}) - h_{\theta_z}(y_i^{(k)}))} \end{aligned} \tag{6}$$

$$\frac{\partial l(\theta, w)}{\partial w_{zk}} = \sum_k Q_k(z) u_j^{(k)} - \sum_k \frac{\exp(w_z^T u^{(k)})}{\sum_{z'} \exp(w_{z'}^T u^{(k)})} u_j^{(k)} \tag{7}$$

Figure 2: E and M steps for the Pairwise PLSI algorithm. Here $h_\theta(y)$ is the logistic function.

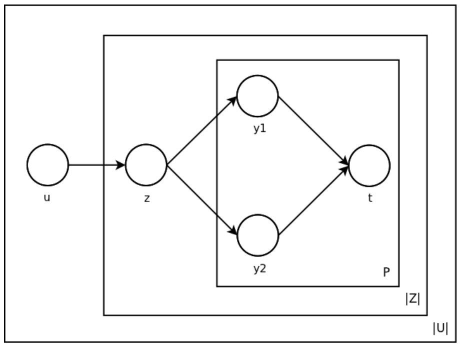


Figure 3: Graphical model describing the relationship between user's pairwise preferences and his/her latent core preferences. Here P is the number of item pairs, $|Z|$ is the number of latent preferences and $|U|$ is the number of users.

In accordance with the above assumptions:

$$P(t|y_i, y_j, u) = \sum_{z \in Z} P(t|y_i, y_j, z)P(z|u)$$

where t is as before, and z is a latent variable representing core preference. Then, for all training pairs, using maximum likelihood estimation,

$$maximize \sum_{k=1}^{M} \log(\sum_z P(t^{(k)}|z, y_i^{(k)}, y_j^{(k)})P(z|u^{(k)}))$$

We discuss modeling choices first, and then describe learning and inference for the model.

5.1 Number of core preferences

The parameter Z controls how much personalized the model is – a Z value of $|U|$ effectively creates a different core preference for each user. There is a tradeoff between accurately modeling a user's preference and model size. A paper describing Google news recommendation [7] followed a comparable approach by assigning a different probability for each user. In the present case, however, that would result in a lot of parameters to learn since the number of parameters in the model is linear in the number of latent preferences. Hence, we consider $|Z|$ to be a small positive number (≤ 8 for our experiments).

5.2 Choosing probability models

We have two model choices to make, $P(t|y_i, y_j, z)$ and $P(z|u)$. For the first, we may use any of the user-agnostic models (latent preferences do not change across users). In the following analysis, we use Logistic Loss as our base model.

$$P(t|y_i, y_j, z; \theta_z) = \frac{h_{\theta_z}(y_j) - h_{\theta_z}(y_i) + 1}{2} \tag{8}$$

$P(z|u)$ represents the relative propensity of users to associate with different latent preferences. We use a softmax-based multinomial function to represent this probability.

$$P(z|u) = \frac{\exp(w_z^T u)}{\sum_{z'} exp(w_{z'}^T u)}$$

5.3 Model fitting

We use Expectation-Maximization to fit the parameters to the model, assuming z as the hidden random variable. Thus, we maximize the likelihood:

$$maximize \sum_{k=1}^{m} \log(P(t|y_i, y_j, u))$$

$$maximize \sum_{k=1}^{m} \log(\sum_z P(t, z|y_i, y_j, u))$$

Let Q_z be an arbitrary distribution over z, for each kth training example $< u, y_i, y_j >$. Then, the E step yields:

$$Q_k(z) = \frac{P(t^{(k)}|y_i^{(k)}, y_j^{(k)}, z)P(z|u^{(k)})}{\sum_{z'} P(t^{(k)}|y_i^{(k)}, y_j^{(k)}, z')P(z'|u^{(k)})}$$

And the M-step becomes:

$$\theta := argmax_\theta \sum_k \sum_{z^{(k)}} Q_k(z^{(k)}) \log(\frac{P(t^{(k)}, z^{(k)}; \theta|y_i, y_j, u)}{Q_k(z^{(k)})})$$

$$\theta := argmax_\theta \sum_k \sum_{z^{(k)}} Q_k(z^{(k)}) \log(P(t^{(k)}; \theta|y_i, y_j, z^{(k)})P(z^{(k)}|u))$$

Substituting the Logistic Loss and the multinomial models, we get the equivalent equations shown in Figure 2. Unlike [8], for the pairwise case, we do not get a closed form expression in the M-step. Hence, we use gradient descent to compute the maximization in each M-step, as shown in Figure 2.

5.4 Inference

Once we have fitted the parameters, we can compute the output class of a pair as follows:

$$P(t|y_i, y_j, u) = \sum_{z \in Z} P(t|y_i, y_j, z)P(z|u)$$

To generate recommendations, we would need a way to rank the communities for a user. It turns out that we get a simple metric for the logistic loss base model, that can be used to rank the recommendations. This becomes possible due to a special property of the logistic and multinomial functions. We present it next.

THEOREM 1. *Let $P(t|u, y_i, y_j)$ be the probability that y_j is preferred over y_i, given by*

$$P(t|y_i, y_j, u) = \sum_{z \in Z} P(t|y_i, y_j, z)P(z|u)$$

where $P(t|y_i, y_j)$ is a logistic function, and $P(z|u)$ is a multinomial distribution. Then $A_{y_k} = \sum_z h_{\theta_z}(y_k)P(z|u)$ is a valid ranking function.

PROOF. Let us assume there exist two items y_i, y_j for which $P(t|u, y_i, y_j)$ is computed. We further assume that y_j is preferred over y_i iff $P(t|u, y_i, y_j) > 0.5$. Define two function values A_1 and A_2, of A_y function.

$$A_1 = \sum_z h_{\theta_z}(y_i)P(z|u), A_2 = \sum_z h_{\theta_z}(y_j)P(z|u)$$

We will proceed to show that $A_2 > A_1 \Leftrightarrow P(t|u, y_i, y_j) > 0.5$ by showing $A_2 - A_1 = 2P(t|u, y_i, y_j) - 1$. Consider

$$A_2 - A_1 = \sum_z h_{\theta_z}(y_j)P(z|u) - \sum_z h_{\theta_z}(y_i)P(z|u)$$
$$= \sum_z P(z|u)(h_{\theta_z}(y_j) - h_{\theta_z}(y_i))$$

Using Equation 8,

$$A_2 - A_1 = \sum_z P(z|u)(2P(t|y_i, y_j, z) - 1)$$
$$= 2(\sum_z P(z|u)P(t|y_i, y_j, z)) - 1$$
$$= 2P(t|u, y_i, y_j) - 1$$

Hence proved. □

6. EVALUATION

We now turn to the evaluation of our proposed models. We first compare the time complexity and scalability of the algorithms, followed by offline evaluation, and finally an online evaluation on linkedin.com.

6.1 Complexity and scalability

We analyze the time complexity for two major operations – model learning (offline), and recommendation computation (online). In a typical scenario, a learned model is used to recompute recommendations frequently for all the users, to utilize new user feedback. But because model learning is not done so frequently, a model with low online complexity and reasonable offline complexity is acceptable.

If L is the size of the recommendation list shown to a user, and J is the total number of group joins, then the number of pairs P generated is $O(L * J)$. For both pairwise models, one step of the gradient descent takes $O(F * P)$, where F is the number of features. In contrast, Pairwise PLSI model is slower to learn, since learning is based on EM iterations. Each E step takes $O(Z * P)$ time, and each M-step consists of another minimization using gradient descent, which is $O(Z * P^2)$.

For non-latent pairwise models, it takes constant time ($O(F)$) to compute ranking score for each user and item. For Pairwise PLSI model, the corresponding time complexity is $O(Z * F)$, where F is the number of features. Thus, ranking for each item is fast in all three algorithms, provided Z is small (2-8).

The parametric models also lend themselves to parallelized rank computation. Given the same model file, different nodes can compute recommendation lists for different users independently. Thus, we can easily scale recommendation computation for millions of users.

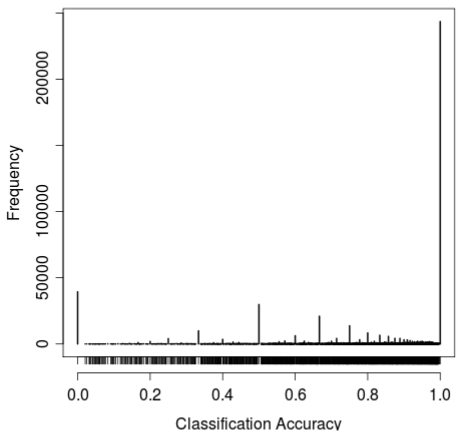
6.2 Offline evaluation

We collected log data for group recommendation over the summer of 2012. The log entries were converted to pairs as described in Section 3. Table 1 shows some statistics for the data. The number of group-pairs are 13 per user on average. For prediction, the test data was selected from logs beginning from a later date than the training data. We chose a relatively small train dataset to speed up model fitting as the learning gains from a bigger dataset were not substantial.

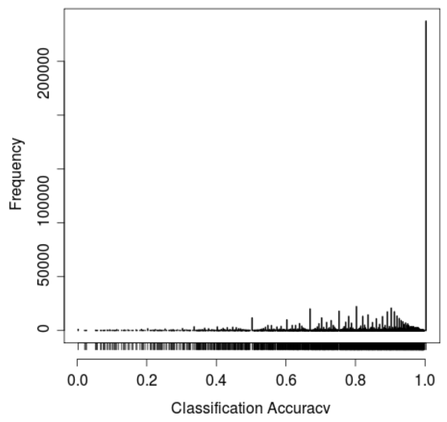
The baseline is a logistic regression model that does not use pairs for learning, as represented in Equation 1. We compare the two base pairwise models, Feature Difference and Logistic Loss, and three versions of the Pairwise PLSI model, with $Z \in \{2, 4, 8\}$.

	Train	Test
Number of users	262K	11.2M
Number of group-pairs	3.4M	NA
Number of group-joins	NA	31.6M

Table 1: Summary statistics for the data collected. Since our ranking function does not require pairs (learning does), we do not compute group-pairs for the test set.



(a) Feature Difference



(b) Pairwise PLSI (z=2)

Figure 4: Per User Accuracy for the best performing non-latent model, and the Pairwise PLSI preferences model (z=2). Pairwise PLSI leads to higher number of users with PUA >0.5. Note the large number of 1's in both algorithms is due to the fact that a lot of users had only 1-2 group joins in the dataset.

Algorithm	Accuracy
Baseline	74.5
Feature Difference	79.4
Logistic Loss	75.4
Pairwise PLSI ($Z = 2$)	83.2
Pairwise PLSI ($Z = 4$)	87.1
Pairwise PLSI ($Z = 8$)	88.6

Table 2: Accuracy on the pair classification task. All pairwise models perform better than the baseline. Latent Preference models deliver a higher accuracy overall.

Learning pairwise preferences. We first evaluate how well the models learn user preference within a pair of groups, by framing it as a classification task. For each pair of groups (y_1, y_2) in the dataset, we classify the pair as positive (1) if y_2 is more preferred than y_1, and zero (0) otherwise. For all models, 5-fold cross validation is used during training. We report accuracies of different models on the pair classification task and compare them in Table 2.

Among non-latent models, Feature Difference delivers the highest accuracy for pair classification, but both pairwise models beat the baseline. Latent models perform better than the other algorithms. Performance of Pairwise PLSI increases with the number of latent preferences, although the relative boost decreases with z. We next see how the accuracy varies for different users.

Per user accuracy. One of the goals of recommendation is user satisfaction, and hence we desire a model that delivers "good" recommendation to all users. For our purposes, we define "good" to be the fraction of group-pairs for a user in the test set that are correctly classified. More precisely,

$$PUA_u = \frac{\textit{Correctly Classified Pairs for user u}}{\textit{Total test pairs for user u}}$$

Figure 4 shows frequency plots of the per user accuracy of the algorithm in the group-pair classification task, for Feature Difference and Pairwise PLSI($Z = 2$). We see that the number of users with non-zero correct classification increase with the Pairwise PLSI model. Also, the number of users with $PUA > 0.5$ is higher for Pairwise PLSI. This suggests that the Pairwise PLSI gives an overall better recommendation experience per user.

Algorithm	Top-5	Top-10	Top-25
Baseline	17.32	25.21	35.72
Feature Difference	17.81	26.25	37.78
Logistic Loss	17.39	25.56	37.81
Pairwise PLSI ($Z = 2$)	18.93	27.82	39.83
Pairwise PLSI ($Z = 4$)	20.04	28.91	41.82

(a) Test set: All groups

Algorithm	Top-5	Top-10	Top-25
Baseline	5.45	8.01	11.25
Feature Difference	5.48	8.03	10.87
Logistic Loss	5.06	7.39	10.54
Pairwise PLSI ($Z = 2$)	5.80	8.40	11.26
Pairwise PLSI ($Z = 4$)	5.99	8.75	11.66

(b) Test set: Groups that have never been recommended to the user during the training period

Table 3: Number of top-k recommendations that were also joined by users during the test period, compared for different models and under two versions of the test set. All reported numbers are in millions. Pairwise PLSI model retrieves more accurate top-k results for both test sets (15% and 10% boost over the baseline for the top-5 metric). For non-latent pairwise models, Feature Difference performs slightly better than the baseline for both test sets, Logistic Loss only for case (a).

Top-k recommendation. We now evaluate the algorithms on the list of groups returned as recommendation. We choose top-k metric for comparing the different models, with $k \in \{5, 10, 25\}$. For any k, the top-k metric is defined as the number of groups among the highest ranked k recommendations that are also joined by the user in the test data. $k = 5$ and $k = 10$ represent the number of group recommendations

Algorithm	Avg. CTR
Baseline	0.0204
Feature Difference	0.0214 (5%)
Logistic Loss	0.0210 (3%)

Table 4: Comparative performance of non-latent models, when deployed on the LinkedIn website. Numbers in parenthesis denote relative improvement over the baseline. Feature Difference delivers a 5% higher click-through-rate (CTR) than the baseline algorithm in production.

typically shown on the homepage, and on the groups page of LinkedIn respectively.

Generating a test set for evaluation is non-trivial due to presentation bias [2]. Since the baseline algorithm also generated recommendations for the users during the test period, there might be a bias in the groups that a user gets exposed to for joining. In one scenario, we could collect all group joins and label them as positive examples. In an alternative overcompensatory scenario, we remove all group joins that happened because of a recommendation and consider the rest as the examples to test on. These are groups that are joined by 'organic' traversal of the website, and hence are a closer estimate of user's preference. We use both types of test sets for evaluation.

Tables 3a and 3b show the top-k metric for the models, under the two versions of the test set. As expected, the number of correctly retrieved groups increases with the increase in k for all models. We observe that Feature Difference again performs better than Logistic Loss among the base pairwise models, but the difference is only slight. The Pairwise PLSI model outperforms all the others, offering a more than 10% boost over the baseline top-5 metric for both test scenarios.

6.3 Online Evaluation

To test whether offline evaluation translates to on-site performance too, we compare the performances of pairwise models after deploying them on the LinkedIn website. We ran the logistic loss and feature difference models to a 5% subset of total LinkedIn users respectively, in addition to the current baseline model, for a period of 2 weeks. We evaluate the click-through-rate (CTR) of the recommendations. The pairwise models perform slightly better, as predicted by the offline analysis. In particular, Feature Difference reports a lift of 5% on the CTR metric (Table 4).

Thus, our results suggest that there is value in learning from pairwise feedback. Although the performance boost using only base pairwise algorithms is slight, Pairwise PLSI model performs better on both top-k and classification metrics. However, the tradeoff is in the learning time of the model, which is considerably higher for Pairwise PLSI.

7. CONCLUSION

As users interact with online recommender systems more and more, understanding user feedback has become important for recommendation quality. We presented our results of applying pairwise learning to community recommendation. Our experiment results show that pairwise models directly adapted from a baseline recommendation model perform better than the baseline non-pairwise model, although only slightly. We argue that the nature of community recommendation does not lend itself well to pairwise comparision due to vast different user preferences towards communities. To that end we proposed a probabilistic latent preference model (Pairwise PLSI) which assumes a set of inherent preferences within each user. Offline evaluation shows that the Pairwise PLSI model significantly outperforms other models.

Acknowledgments The authors would like to thank Dan Cosley, for feedback on initial drafts of the paper, and LinkedIn for providing the data used in the experiments.

8. REFERENCES

[1] S. Balakrishnan and S. Chopra. Two of a kind or the ratings game? Adaptive pairwise preferences and latent factor models. In *Proc. ICDM*, 2010.

[2] J. Bar-Ilan, K. Keenoy, M. Levene, and E. Yaari. Presentation bias is significant in determining user preference for search results-A user study. *American Soc. for Inf. Science and Tech.*, 2009.

[3] Z. Cao, T. Qin, T.-Y. Liu, M.-F. Tsai, and H. Li. Learning to rank: from pairwise approach to listwise approach. In *Proc. ICML*, 2007.

[4] W.-Y. Chen, J.-C. Chu, J. Luan, H. Bai, Y. Wang, and E. Y. Chang. Collaborative filtering for orkut communities: discovery of user latent behavior. In *Proc. WWW*, 2009.

[5] W.-Y. Chen, D. Zhang, and E. Y. Chang. Combinational collaborative filtering for personalized community recommendation. In *Proc. KDD*, 2008.

[6] N. Craswell, O. Zoeter, M. Taylor, and B. Ramsey. An experimental comparison of click position-bias models. In *Proc. WSDM*, 2008.

[7] A. S. Das, M. Datar, A. Garg, and S. Rajaram. Google news personalization: scalable online collaborative filtering. In *Proc. WWW*, 2007.

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

[9] T. Joachims. Optimizing search engines using clickthrough data. In *Proc. KDD*, 2002.

[10] T. Joachims, L. Granka, B. Pan, H. Hembrooke, F. Radlinski, and G. Gay. Evaluating the accuracy of implicit feedback from clicks and query reformulations in web search. *ACM Trans. Inf. Syst.*, 2007.

[11] T. Kamishima. Nantonac collaborative filtering: recommendation based on order responses. In *Proc. KDD*, 2003.

[12] H. Li. Learning to rank for information retrieval and natural language processing. *Synthesis Lectures on Human Language Tech.*, 2011.

[13] N. N. Liu, M. Zhao, and Q. Yang. Probabilistic latent preference analysis for collaborative filtering. In *Proc. CIKM*, 2009.

[14] T.-Y. Liu. *Learning to rank for information retrieval.* Springerverlag Berlin, 2011.

[15] R. Pan and M. Scholz. Mind the gaps: weighting the unknown in large-scale one-class collaborative filtering. In *Proc. KDD*, 2009.

[16] R. Pan, Y. Zhou, B. Cao, N. N. Liu, R. Lukose, M. Scholz, and Q. Yang. One-class collaborative filtering. In *Proc. ICDM*, 2008.

[17] J.-F. Pessiot, T.-V. Truong, N. Usunier, M.-R. Amini, and P. Gallinari. Learning to rank for collaborative filtering. In *ICEIS (2)'07*, 2007.

[18] K. Radinsky and N. Ailon. Ranking from pairs and triplets: information quality, evaluation methods and query complexity. In *Proc. WSDM*, 2011.

[19] P. Ray. Independence of irrelevant alternatives. *Econometrica: Journal of the Econometric Soc.*, 1973.

[20] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *Proc. UAI*, 2009.

[21] E. Spertus, M. Sahami, and O. Buyukkokten. Evaluating similarity measures: a large-scale study in the orkut social network. In *Proc. KDD*, 2005.

[22] N. Zheng, Q. Li, S. Liao, and L. Zhang. Which photo groups should i choose? a comparative study of recommendation algorithms in flickr. *J. Inf. Sci.*, 2010.