Personalized Recommendation of User Comments via Factor Models

Deepak Agarwal **Bee-Chung Chen** **Bo Pang**
Yahoo! Research
701 First Ave
Sunnyvale, CA 94089
{dagarwal,beechun,bopang}@yahoo-inc.com

Abstract

In recent years, the amount of user-generated opinionated texts (e.g., reviews, user comments) continues to grow at a rapid speed: featured news stories on a major event easily attract thousands of user comments on a popular online News service. How to consume subjective information of this volume becomes an interesting and important research question. In contrast to previous work on review analysis that tried to filter or summarize information for a generic *average* user, we explore a different direction of enabling personalized recommendation of such information.

For each user, our task is to rank the comments associated with a given article according to personalized user preference (i.e., whether the user is likely to like or dislike the comment). To this end, we propose a factor model that incorporates rater-comment and rater-author interactions simultaneously in a principled way. Our full model significantly outperforms strong baselines as well as related models that have been considered in previous work.

1 Introduction

Recent years have seen rapid growth in user-generated opinions online. Many of them are user reviews: a best-seller or a popular restaurant can get over 1000 reviews on top review sites like Amazon or Yelp. A large quantity of them also come in the form of user comments on blogs or news articles. Most notably, during the short period of time for which a major event is active, news stories on one single event can easily attract over ten thousand comments on a popular online news site like Yahoo! News. One question becomes immediate: how can we help people consume such gigantic amount of opinionated information?

One possibility is to take the summarization route. Briefly speaking (see Section 2 for a more detailed discussion), previous work has largely formulated review summarization as automatically or manually identify ratable aspects, and present overall sentiment polarity for each aspect (Hu and Liu, 2004; Popescu and Etzioni, 2005; Snyder and Barzilay, 2007; Titov and McDonald, 2008). A related line of research looked into predicting helpfulness of reviews in the hope of promoting those with better quality, where helpfulness is usually defined as some function over the percentage of users who found the review to be helpful (Kim et al., 2006; Liu et al., 2007; Danescu-Niculescu-Mizil et al., 2009). In short, the focus of previous work has been on distilling subjective information for an *average* user.

Whether opinion consumers are looking for quality information or just wondering what other people think, each may have different purposes or preferences that is not well represented by a generic average user. If we think about how we deal with information content overflow on the Web, there have been two main frameworks to identify relevant information for each person. One is search. Indeed many top review sites allow users to search within reviews for a given entity. But this is only useful when users have explicit information needs that can be formulated as queries. The other paradigm is recommendation: based on what users have liked or disliked in the past, the system will automatically recommend

Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pages 571–582,
Edinburgh, Scotland, UK, July 27–31, 2011.

new items.

Can we provide similar recommendation mechanisms to help users consume large quantities of subjective information? Many commenting environments allow users to mark "like" or "dislike" over existing comments (e.g., Yahoo! News comments, Facebook posts, or review sites that allow helpfulness votes). Can we learn from users' past preferences, so that when a user is reading a new article, we have a system that automatically ranks its comments according to their likelihood of being liked by the user? This can be used directly to create personalized presentation of comments (e.g., into a "like" column and a "dislike" column), as well as enabling down-stream applications such as personalized summarization.

Recommending textual information has recently attracted more attention. So far, the focus has been mainly on recommending news articles (Ahn et al., 2007; Das et al., 2007). Our task differs in several aspects. Intuitively, recommending news articles is largely about identifying the topics of interest to a given user, and it is conceivable that unigram representation of full-length articles can reasonably capture that information. In our case, most comments for an article a user is reading are already of interest to that user topically. Which ones the user ends up liking may depend on several non-topical aspects of the text: whether the user agrees with the viewpoint expressed in the comment, whether the comment is convincing and well-written, etc. Previous work has shown that such analysis can be more difficult than topic-based analysis (Pang and Lee, 2008), and we have the additional challenge that comments are typically much shorter than full-length articles. However, the difficulty in analyzing the textual information in comments can be alleviated by additional contextual information such as author identities. If between a pair of users one consistently likes or dislikes the other, then at least for the heavy users, this authorship information alone could be highly informative. Indeed, previous work in collaborative filtering has usually found no additional gain from leveraging content information when entity-level preference information is abundant.

In this paper, we present a principled way of utilizing multiple sources of information for the task of recommending user comments, which significantly outperforms strong baseline methods, as well as previous methods proposed for text recommendation. While using authorship information alone tends to provide stronger signal than using textual information alone, to our surprise, even for heavy users, adding textual information to the authorship information yields additional improvements.

2 Related Work

There are two main bodies of related work: our problem formulation is closer to collaborative filtering, while the nature of the text we are dealing with has more in common with opinion mining and sentiment analysis.

Our approach is related to a large body of work in collaborative filtering. While a proper survey is not possible here, we describe some of the approaches that are germane. Classical approaches in collaborative filtering are based on item-item/user-user similarity, these are nearest-neighbor methods where the response for a user-item pair is predicted based on a local neighborhood mean (Sarwar et al., 2001; Wang et al., 2006). In general, neighborhoods are defined by measuring similarities between users/items through correlation measures like Pearson, cosine similarities, etc. Better approaches to estimate similarities have also been proposed in Koren (2010). However, modern methods based on matrix factorization have been shown to outperform nearest neighbor methods (Salakhutdinov and Mnih, 2008a,b; Bell et al., 2007). Generalizations of matrix factorization to include both features and past ratings have been proposed (Agarwal and Chen, 2009; Stern et al., 2009). The approach in this paper is an extension where in addition to interactions among users and items (comments in our case), we also consider the authorship information. Three-way interactions were recently studied for personalized tag recommendation (Rendle and Lars, 2010). Their model was based on the sum of two-way interactions, and was trained by using pairwise tag preferences for each (user, item) pair. However, no features were considered, which is an important consideration for us. We show using both text and authorship provides the best performance.

Our work is also related to news personalization that has received increasing attention in the last few

years. For instance, Billsus and Pazanni (2007) describes an approach to build user profile models for adaptive personalization in the context of mobile content access. Their approach is based on a hybrid model that combines content-based approaches with similarity methods used in recommender systems. This is further exemplified in the work by Ahn et al. (2007) where text processing techniques are used to build content profiles for users to recommend personalized news. In our experiments, we show that such approaches are inferior to our method. A content agnostic approach based on collaborative filtering techniques was proposed by Das et al. (2007); cold-start for new items/users was not their focus, but is important for our task — candidate comments for recommendation are often not in training data.

As discussed in Section 1, previous work in opinion mining and sentiment analysis has addressed the information consumption challenge via review summarization. Discussion of early work in that direction can be found in Pang and Lee (2008). In this line of work, opinions for each given aspect are usually summarized as the average sentiment polarity associated with that aspect. Related to that, people have looked into predicting review helpfulness given the textual information in reviews, where helpfulness is either defined as the percentage of users who have voted the review to be helpful (Kim et al., 2006), or labeled by annotators according to a set of criteria (Liu et al., 2007). Our goal differs in that we look for personalized ranking (what a specific user might like) rather than generic quality (what an average user might like). Subsequently, there has been work that tried to predict similarly defined helpfulness scores using meta-information over the reviewer. For instance, whether the author has used his/her true name or where the user is from (Danescu-Niculescu-Mizil et al., 2009), as well as graph structure in the social network between reviewers (Lu et al., 2010). In this work, we simply use author identity to provide more context to the short text; in future work, additional meta-information over users can easily be incorporated via our model.

As discussed in Section 1, whether a rater likes a comment or not may depend on whether they agree with the viewpoint expressed in the text and quality of the text. While previous work has not looked into the reader-comments relationship, there has been related work on identifying political orientations or viewpoints (Lin and Hauptmann, 2006; Lin et al., 2006; Mullen and Malouf, 2006, 2008; Laver et al., 2003); whether a piece of text expresses support or opposition in congressional debates (Thomas et al., 2006) or online debates (Somasundaran and Wiebe, 2009, 2010); as well as identifying contrastive relationship (Kawahara et al., 2010). Note that it is not trivial to use previous work along this line to directly serve as sub-components in our setting. For instance, for work on identifying political orientations or viewpoints, the training data consists of text with the desired labels. In our setting, our labels come in the form of whether users liked or disliked a previous comment. In the simplest case, we might have pair-wise constraints on whether two pieces of text have the same viewpoints (i.e., liked or disliked by the same rater), which would yield a different learning problem akin to the metric learning problem; note, however, the complication that two pieces of text receiving different labels from a given user might not necessarily contain contrasting viewpoints. Consequently, rather than trying to reduce this problem to a set of known text classification tasks, we address this task via a collaborative filtering framework that incorporates textual features.

3 Method

In this section, we describe our model that predicts rater affinity to comments. A key strength of our model is the ability to incorporate rater-comment and rater-author interactions simultaneously in a principled fashion. Our model also provides a seamless mechanism to transition from cold-start (where recommendations need to be made for users or items with no or few past ratings) to warm-start scenarios — with a large amount of data, it fits a per-rater (author) model; with increase in data sparsity, the model applies a small sample size correction through features (in our case, textual features). The exact formula for such corrections in the presence of sparsity is based on parameter estimates that are obtained by applying an EM algorithm to the training data.

3.1 Model

Notation: Let y_{ij} denote the rating that user i, called the *rater*, gives to comment j. Since throughout, we use suffix i to denote a rater and suffix j to denote a comment, we slightly abuse notation and let $\boldsymbol{x}_i$ (of dimension p_u) and $\boldsymbol{x}_j$ (of dimension p_c) denote feature vectors of user i and comment j respectively. For example, $\boldsymbol{x}_i$ can be the bag of words representation (a sparse vector) inferred through text analysis on comments voted positively by user i in the past, and $\boldsymbol{x}_j$ can be the bag of words representation for comment j. We use $a(j)$ to denote the author of comment j, and use μ_{ij} to denote the mean rating by rater i on comment j, i.e., $\mu_{ij} = E(y_{ij})$. Of course it is impossible to estimate μ_{ij} empirically since each user i usually rates a comment j at most once.

Model specification: We work in a generalized linear model framework (McCullagh and Nelder, 1989) that assumes μ_{ij} (or some monotone function h of μ_{ij}) is an additive function of (1) the rater bias α_i of user i since some users may have a tendency of rating comments more positively or negatively than others, (2) popularity β_j of comment j, which could reflect the quality of the comment in this setting, and (3) the author reputation $\gamma_{a(j)}$ of user $a(j)$ since comments by a reputed author may in general get more positive ratings. Thus, the overall bias is $\alpha_i + \beta_j + \gamma_{a(j)}$.

In addition to the bias, we include terms that capture interactions among entities (raters, authors, comments). Indeed, capturing such interactions is a non-trivial part of our modeling procedure. In our approach, we take recourse to *factor* models that have been widely used in collaborative filtering applications in recent times. The basic idea is to attach latent factors to each rater, author and comment. These latent factors are finite dimensional Euclidean vectors that are unknown and estimated from the data. They provide a succinct representation of various *aspects* that are important to explain interaction among entities. In our case, we use the following factors — (a) user factor $\boldsymbol{v}_i$ of dimension $r_v(\geq 1)$ to model rater-author affinity, (b) user factor $\boldsymbol{u}_i$ and comment factor $\boldsymbol{c}_j$ of dimension $r_u(\geq 1)$ to model rater-comment affinity. Intuitively, each could represent viewpoints of users or comments along different dimensions.

i	index for raters
j	index for comments
$a(j)$	author of comment j
y_{ij}	rating given by rater i to comment j
μ_{ij}	mean rating given by rater i to comment j
$\boldsymbol{x}_j$	feature vector of comment j (e.g., textual features in comment j)
$\boldsymbol{x}_i$	feature vector of user i (e.g., comments voted positively by user i)
bias terms:	
α_i	rater bias of user i
β_j	popularity of comment j (e.g., quality of the comment)
$\gamma_{a(j)}$	reputation of the author of comment j
interaction terms:	
$\boldsymbol{v}_i$	user factor for rater-author affinity
$\boldsymbol{u}_i$, $\boldsymbol{c}_j$	factors for rater-comment affinity

Table 1: Table of Notations.

Affinity of rater i to comment j by author $a(j)$ is captured by (1) similarity between viewpoints of users i and $a(j)$, measured by $\boldsymbol{v}_i'\boldsymbol{v}_{a(j)}$; and (2) similarity between the preferences of user i and the perspectives reflected in comment j, measured by $\boldsymbol{u}_i'\boldsymbol{c}_j$. The overall interaction is $\boldsymbol{v}_i'\boldsymbol{v}_{a(j)} + \boldsymbol{u}_i'\boldsymbol{c}_j$. Then, the mean rating μ_{ij}, or more precisely $h(\mu_{ij})$, is modeled as the sum of bias and interaction terms. Mathematically, we assume:

$$\begin{aligned} y_{ij} &\sim N(\mu_{ij},\ \sigma_y^2) \text{ or Bernoulli}(\mu_{ij}) \\ h(\mu_{ij}) &= \alpha_i + \beta_j + \gamma_{a(j)} + \boldsymbol{v}_i'\boldsymbol{v}_{a(j)} + \boldsymbol{u}_i'\boldsymbol{c}_j \end{aligned} \quad (1)$$

For numeric ratings, we use the Gaussian distribution denoted by N(mean,var); for binary ratings, we use the Bernoulli distribution. For Gaussian, $h(\mu_{ij}) = \mu_{ij}$, and for Bernoulli, we assume $h(\mu_{ij}) = \log \frac{\mu_{ij}}{1-\mu_{ij}}$, which is the commonly used logistic transformation.

Table 1 summarizes the notations for easy references. We denote the full model specified above as **vv+uc** since both user-user interaction $\boldsymbol{v}_i'\boldsymbol{v}_{a(j)}$ and user-comment interaction $\boldsymbol{u}_i'\boldsymbol{c}_j$ are modeled at the same time.

Latent factors: A natural approach to estimate latent factors in Equation 1 is through a maximum likelihood estimation (MLE) approach. This does

not work in our scenario since a large fraction of entities have small sample size. For instance, if a comment is rated only by one user and $r_u > 1$, the model is clearly overparametrized and the MLE of the comment factor would tend to learn idiosyncrasies in the training data. Hence, it is imperative to impose constraints on the factors to obtain estimates that generalize well on unseen data. We work in a Bayesian framework where such constraints are imposed through prior distributions. The crucial issue is the selection of appropriate priors. In our scenario, we need priors that provide a good *backoff* estimate when interacting entities have small sample sizes. For instance, to estimate latent factors of a user with little data, we provide a backoff estimate that is obtained by *pooling* data across users with the same user features. We perform such a pooling through regression, the mathematical specification is given below.

$$\begin{aligned} \alpha_i &\sim N(g'\boldsymbol{x}_i,\ \sigma_\alpha^2), & \boldsymbol{u}_i &\sim N(G\boldsymbol{x}_i,\ \sigma_u^2), \\ \beta_j &\sim N(d'\boldsymbol{x}_j,\ \sigma_\beta^2), & \boldsymbol{c}_j &\sim N(D\boldsymbol{x}_j,\ \sigma_c^2), \\ \gamma_{a(j)} &\sim N(0,\ \sigma_\gamma^2), & \boldsymbol{v}_i &\sim N(\mathbf{0},\ \sigma_v^2), \end{aligned}$$

where $g^{p_u \times 1}$ and $d^{p_c \times 1}$ are regression weight vectors, and $G^{r_u \times p_u}$ and $D^{r_u \times p_c}$ are regression weight matrices. These regression weights are learnt from data and provide the backoff estimate. Take the prior distribution of $\boldsymbol{u}_i$ for example. We can rewrite the prior as $\boldsymbol{u}_i = G\boldsymbol{x}_i + \boldsymbol{\delta}_i$, where $\boldsymbol{\delta}_i \sim N(\mathbf{0},\ \sigma_u^2)$. If user i has no rating in the training data, $\boldsymbol{u}_i$ will be predicted as the prior mean (backoff) $G\boldsymbol{x}_i$, a linear projection from the feature vector $\boldsymbol{x}_i$ through matrix G learnt from data. This projection can be thought of as a multivariate linear regression problem with weight matrix G, one weight vector per dimension of $\boldsymbol{u}_i$. However, if user i has many ratings in the training data, we will precisely estimate the per-user *residual* $\boldsymbol{\delta}_i$ that is not captured by the regression $G\boldsymbol{x}_i$. For sample sizes in between these two extremes, the per user residual estimate is "shrunk" toward zero — amount of shrinkage depends on the sample size, past user ratings, variability in ratings on comments rated by the user, and the value of variance components $\sigma_\cdot^2$s.

3.2 Special Cases of Our Model

Our full model (**vv+uc**) includes several existing models explored in collaborative filtering and social networks as special cases.

The matrix factorization model: This model assumes the mean rating of user i on item j is given by $h(\mu_{ij}) = \alpha_i + \beta_j + \boldsymbol{u}_i'\boldsymbol{c}_j$, and the mean of the prior distributions on $\alpha_i, \beta_j, \boldsymbol{u}_i, \boldsymbol{c}_j$ are zero, i.e., $g = d = G = D = \mathbf{0}$. Recent work clearly illustrates that this method obtains better predictive accuracy than classical collaborative filtering techniques based on item-item similarity (Bell et al. (2007)).

The **uc** *model:* This is also a matrix factorization model but with priors based on regressions (i.e., non-zero g, d, G, D). It provides a mechanism to deal with both cold and warm-start scenarios in recommender applications (Agarwal and Chen (2009)).

The **vv** *model:* This model assumes $h(\mu_{ij}) = \alpha_i + \gamma_{a(j)} + \boldsymbol{v}_i'\boldsymbol{v}_{a(j)}$. It was first proposed by Hoff (2005) to model interactions in social networks. The model was fitted to small datasets (at most a few hundred nodes) and the goal was to test certain hypotheses on social behavior, out-of-sample prediction was not considered.

The low-rank **bilinear** *regression model:* Here, $h(\mu_{ij}) = g'\boldsymbol{x}_i + d'\boldsymbol{x}_j + \boldsymbol{x}_i'G'D\boldsymbol{x}_j$. This is a regression model purely based on features with no per-user or per-comment latent factors. In a more general form, $\boldsymbol{x}_i'G'D\boldsymbol{x}_j$ can be written as $\boldsymbol{x}_i'A\boldsymbol{x}_j$, where $A^{p_u \times p_c}$ is the matrix of regression weights (Chu and Park, 2009). However, since $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are typically high dimensional, A can be a large matrix that needs to be learnt from data. To reduce dimensionality, one can decompose A as $A = G'D$, where the number of rows in D and G are small. Thus, instead of learning A, we learn a low-rank approximation of A. This ensures scalability and provides an attractive method to avoid over-fitting.

3.3 Model Fitting

Model fitting for our model is based on the expectation-maximization (EM) algorithm (Dempster et al., 1977). For ease of exposition and space constraints, we only provide a sketch of the algorithm for the Gaussian case, the logistic model can be fitted along the same lines by using a variational approximation (see Agarwal and Chen (2009)).

Let $\boldsymbol{Y} = \{y_{ij}\}$ denote the set of the observed ratings. In the EM parlance, this is "incomplete"

data that gets augmented with the latent factors $\boldsymbol{\Theta} = \{\boldsymbol{u}_i, \boldsymbol{v}_i, \boldsymbol{c}_j\}$ to obtain the "complete" data. The goal of the EM algorithm is to find the parameter $\boldsymbol{\eta} = (g, d, G, D, \sigma_\alpha^2, \sigma_\beta^2, \sigma_u^2, \sigma_v^2, \sigma_y^2)$ that maximizes the "incomplete" data likelihood $\Pr(\boldsymbol{Y}|\boldsymbol{\eta}) = \int \Pr(\boldsymbol{Y}, \boldsymbol{\Theta}|\boldsymbol{\eta}) d\boldsymbol{\Theta}$ that is obtained after marginalization (taking expectation) over the distribution of $\boldsymbol{\Theta}$. Since such marginalization is not available in closed form for our model, we use the EM algorithm.

EM algorithm: The complete data log-likelihood $l(\boldsymbol{\eta}; \boldsymbol{Y}, \boldsymbol{\Theta})$ for the full model in the Gaussian case (where $h(\mu_{ij}) = \mu_{ij}$) is given by $l(\boldsymbol{\eta}; \boldsymbol{Y}, \boldsymbol{\Theta}) =$

$$
\begin{aligned}
&-\tfrac{1}{2}\textstyle\sum_{ij}\left((y_{ij}-\mu_{ij})^2/\sigma_y^2 + \log\sigma_y^2\right)\\
&-\tfrac{1}{2}\textstyle\sum_{i}\left((\alpha_i - g'\boldsymbol{x}_i)^2/\sigma_\alpha^2 + \log\sigma_\alpha^2\right)\\
&-\tfrac{1}{2}\textstyle\sum_{j}\left((\beta_j - d'\boldsymbol{x}_j)^2/\sigma_\beta^2 + \log\sigma_\beta^2\right)\\
&-\tfrac{1}{2}\textstyle\sum_{i}\left(\|\boldsymbol{u}_i - G\boldsymbol{x}_i\|^2/\sigma_u^2 + r_u\log\sigma_u^2\right)\\
&-\tfrac{1}{2}\textstyle\sum_{j}\left(\|\boldsymbol{c}_j - D\boldsymbol{x}_j\|^2/\sigma_c^2 + r_u\log\sigma_c^2\right),\\
&-\tfrac{1}{2}\textstyle\sum_{i}\left(\boldsymbol{v}_i'\boldsymbol{v}_i/\sigma_v^2 + r_v\log\sigma_v^2 + \gamma_i^2/\sigma_\gamma^2 + \log\sigma_\gamma^2\right),
\end{aligned}
$$

where r_u is the dimension of factors $\boldsymbol{u}_i$ and $\boldsymbol{c}_j$, and r_v is the dimension of $\boldsymbol{v}_i$. Let $\boldsymbol{\eta}^{(t)}$ denote the estimated parameter setting at the t^{th} iteration. The EM algorithm iterates through the following two steps until convergence.

- **E-step:** Compute $f_t(\boldsymbol{\eta}) = E_{\boldsymbol{\Theta}}[l(\boldsymbol{\eta}; \boldsymbol{Y}, \boldsymbol{\Theta}) \,|\, \boldsymbol{\eta}^{(t)}]$ as a function of $\boldsymbol{\eta}$, where the expectation is taken over the posterior distribution of $(\boldsymbol{\Theta} \,|\, \boldsymbol{\eta}^{(t)}, \boldsymbol{Y})$. Note that here $\boldsymbol{\eta}$ is the input variable of function f_t, but $\boldsymbol{\eta}^{(t)}$ consists of known quantities (determined in the previous iteration).
- **M-step:** Find the $\boldsymbol{\eta}$ that maximizes the expectation computed in the E-step.

$$\boldsymbol{\eta}^{(t+1)} = \arg\max_{\boldsymbol{\eta}} f_t(\boldsymbol{\eta})$$

Since the expectation in the E-step is not available in a closed form, we use a Gibbs sampler to compute the Monte Carlo expectation (Booth and Hobert, 1999). The Gibbs sampler repeats the following procedure L times. It samples α_i, γ_i, β_j, $\boldsymbol{u}_i$, $\boldsymbol{v}_j$, and $\boldsymbol{c}_j$ sequentially one at a time by sampling from the corresponding full conditional distributions. The full conditional distributions are all Gaussian, hence they are easy to sample. Once a Monte Carlo expectation is calculated from the samples, an updated estimate of $\boldsymbol{\eta}$ is obtained in the M-step. The optimization of variance components $\sigma_\cdot^2$s in the M-step is available in closed form, the regression parameters are estimated through off-the-shelf linear regression routines. We note that the posterior distribution of latent factors for known $\boldsymbol{\eta}$ is multi-modal, we have found the Monte Carlo based EM method to outperform other optimization methods like gradient descent in terms of predictive accuracy.

4 Experiments

4.1 Data

We obtained comment rating data between March and May, 2010 from Yahoo! News, with all user IDs anonymized. On this site, users can post comments on news article pages and rate the comments made by others through thumb-up (positive) or thumb-down (negative) votes. Clearly, for articles with very few comments, there is no need to recommend comments. Also, we do not expect deep personalized recommendations for users who have rated very few comments in the past. To focus on instances of interest to us, we restricted ourselves to a subset of the rating data associated with relatively heavy raters. In particular, we formed the experimental dataset by randomly selecting 9,003 raters who provided at least 200 ratings (of which at least 10 were positive and 10 were negative), 189,291 authors who received at least 20 ratings, and 5,088 news articles that received at least 40 comments in the raw dataset during the three-month period. Note that the per entity sample size in the experimental dataset can be smaller than the thresholds specified above. For instance, a rater with more than 200 ratings in the raw dataset can have fewer than 200 in the experimental dataset due to the removal of certain authors or news articles. (See Figure 2 for a distribution of users with different activity levels.) In total, we have 4,440,222 ratings on 1,197,098 comments.

The 5,088 news articles were split into training articles (the earliest 50%), tuning articles (next 5%), and test articles (the last 45%) based on their publication time. The ratings and comments were split into training, tuning, and test sets according to the article they were associated with. All tuning parameters are determined using the tuning set, and performances are reported over the test set. Note that

this training-test split ensures that performance on the test data best simulates our application scenarios. It also creates a completely cold-start situation for comments — no comment in the test set has any past rating in the training set.

4.2 Experimental Setup

Features: All comments were tokenized, lowercased, with stopwords and punctuations removed. We limited the vocabulary to the 10K most frequent tokens in all comments associated with the training articles. (See Section 4.3.3 for a discussion on the effect of the vocabulary size.) For a given comment j, $\boldsymbol{x}_j$ is its bag of words representation, L_2 normalized. For term weighting, we experimented with both presence value and tf-idf weighting. The latter gives slight better performance. Rater feature vector $\boldsymbol{x}_i$ is created by summing over the feature vectors of all comments rated positively by rater i, which is then L_2 normalized.

Methods: We compare the following methods based on our model: The full model **vv+uc**, as well as the three main special cases, **vv**, **uc**, and **bilinear**, as defined in Section 3. The dimensions of $\boldsymbol{v}_i$, $\boldsymbol{u}_i$ and $\boldsymbol{c}_j$ (i.e., r_v and r_u), and the rank of **bilinear** are selected to obtain the best AUC on the tuning set. In our experiments, $r_v = 2, r_u = 3$ and rank of **bilinear** is 3. In addition, we also evaluate the following baseline methods that predict per-user preferences in isolation, primarily based on textual information.

- **Cosine similarity (cos):** $\boldsymbol{x}_i'\boldsymbol{x}_j$. This is simply based on how similar a new comment j is to the comments rater i has liked in the past.
- **Per-user SVM (svm):** For each rater, train a support vector machine (SVM) classifier using only comments ($\boldsymbol{x}_j$) rated by that user.
- **Per-user Naive Bayes (nb):** For each rater, train a Naive Bayes classifier using only comments ($\boldsymbol{x}_j$) rated by that user.[1]

Note that SVMs typically yield the best performance on text classification tasks; a Naive Bayes classifier can be more robust over shorter text spans common in user comments given the high variance. For fair comparisons, for the three baseline methods, we use a simple way of utilizing author information: the feature space is augmented with author IDs and each $\boldsymbol{x}_j$ is augmented with $a(j)$[2]. In Section 4.3, we only report results using the augmented feature vectors since they yield better performance (though the difference is fairly small).

Performance metrics: We use two types of metrics to measure the performance of a method: (1) A global metric based on Receiver Operating Characteristic (ROC) and (2) Precision at rank k (P@k). The former measures the overall correlation of predicted scores for a method with the observed ratings in the test set, while the latter measures the performance of a hypothetical top-k recommendation scenario using the method. To summarize an ROC curve into a single number, we use the Area Under the ROC Curve (AUC). Since random guess yields AUC score of 0.5, regardless of the class distribution, using this measure makes it convenient for us to compare the performance over different subsets of the data (where class distributions could be different). The P@k of a method is computed as follows: (1) For each rater, rank comments that the rater rated in the test set according to the scores predicted by the method, and compute the precision at rank k for that rater; and then (2) average the per-rater precision numbers over all raters. To report P@k, for $k = 5, 10, 20$, we only use raters who have at least 50 ratings in the test set. Statistical significance based on a two-sample t-test across raters is also reported.

4.3 Results

4.3.1 Main comparisons

We first show the ROC curves of different methods on the test set in Figure 1, and the AUCs and precisions in Table 2. Results from significance tests are in Table 3.

First, note that while **svm** significantly outperforms random guesses and **nb**, it is worse than **bilinear**, which is also using (mostly) textual information, but learns the model for all users together,

[1] As we mentioned in Section 4.1, not all users have training data of both classes in the experimental dataset. For **svm** and **nb**, we use the following backoff: for users with training data from only c_i, we predict c_i; for users with no training data at all, we predict the majority class, in this case, the positive class.

[2] We assign weight 1 to $a(j)$, so that the author information have the same impact as the textual features.

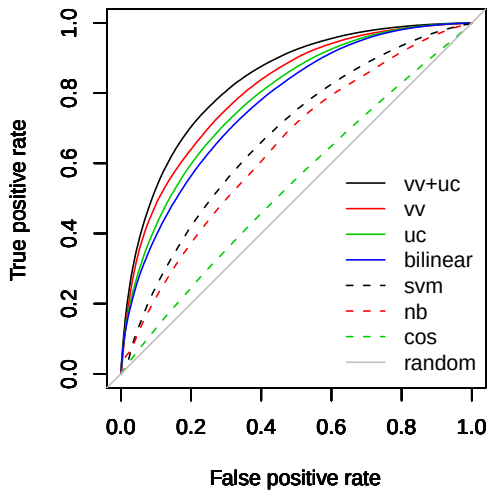


Figure 1: ROC curves of different models

Method	AUC	P@5	P@10	P@20
vv+uc	0.8360	0.9152	0.9079	0.8942
vv	0.8090	0.8810	0.8807	0.8727
uc	0.7857	0.9046	0.8921	0.8694
bilinear	0.7701	0.9028	0.8894	0.8668
svm	0.6768	0.7814	0.7678	0.7497
nb	0.6465	0.7660	0.7486	0.7309
cos	0.5382	0.6834	0.6813	0.6754

Table 2: AUCs and precisions of different models.

rather than in isolation. Next, **uc** outperforms **bilinear** (significantly in AUC, P@10 and P@20), showing per-user and per-comment latent factors help. Note that **vv** outperforms **uc** in ROC, AUC and P@20, but is worse than **uc** in P@5 and P@10; we will take a closer look at this later. Finally, the full model **vv+uc** significantly outperforms both **vv** and **uc**, achieving 0.83 in AUC, and close to 90% in precision at rank 20.

4.3.2 Break-down by user activity level

Next, we investigate model performance in different subsets of the test set. For succinctness, we use AUC as our performance metric. In Figure 2(a), we breakdown model performance by different author activity levels. In Figure 2(b), we breakdown model performance by different voter activity levels. We also generated similar plots with the y-axis replaced by P@5, P@10 and P@20, and observed the same trend except that **vv** starts to outperform **uc** at different user activity thresholds for different metrics.

Comparison	Metrics	p-value
vv+uc $>$ **vv**	All	$< 10^{-7}$
vv+uc $>$ **uc**	All	$< 10^{-20}$
uc $>$ **bilinear**	All except P@5	< 0.006
bilinear $>$ **svm**	All	$< 10^{-20}$
vv $>$ **svm**	All	$< 10^{-20}$
svm $>$ **nb**	All	$< 10^{-8}$
nb $>$ **cos**	All	$< 10^{-20}$

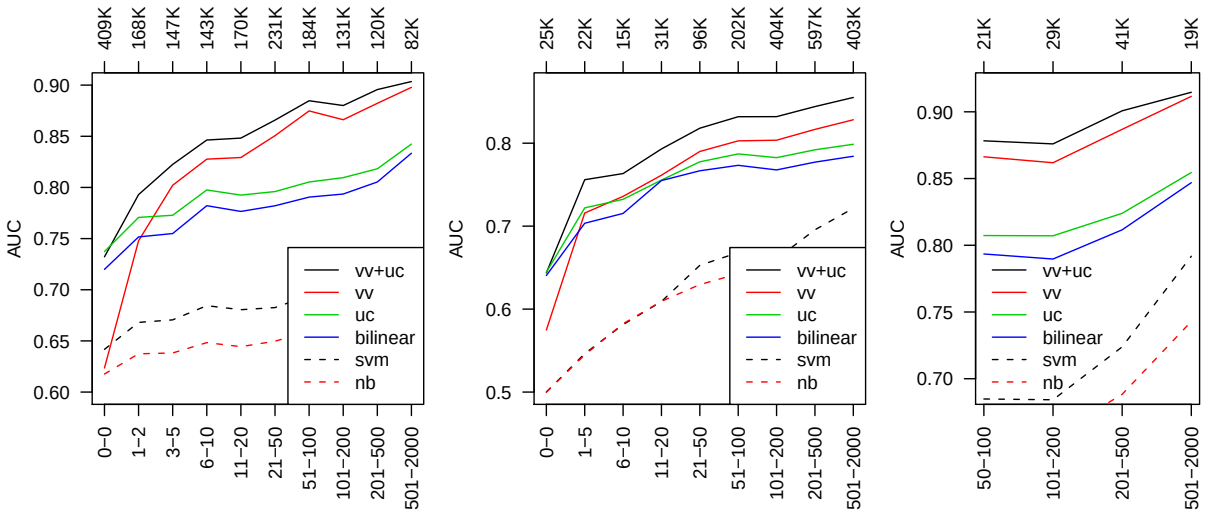
Table 3: Paired t-test results. Note that **uc** is better than **bilinear** in P@5, but not significant. The orders of **uc** and **vv** are not consistent across different metrics.

Not surprisingly, **vv** performs poorly for raters or authors with no ratings observed in the training data. However, once we have a small amount of ratings, it starts to outperform **uc**, even though intuitively, the textual information in the comment should be more informative than the authorship information alone. Using paired t-tests with significance level 0.05, we report when **vv** starts to significantly outperform **uc** in the following table, which is interpreted as follows — **vv** is not significantly worse than **uc** in metric M if the author of a test comment received at least N_{eq} ratings in the training set, and **vv** significantly outperforms **uc** in metric M if the author received at least N_{gt} ratings in the training set.

Metric M	P@5	P@10	P@20	AUC
# Ratings N_{eq}	50	5	5	5
N_{gt}	1000	50	5	5

Recall that our training/test split is by article. Since we have never observed a rater's preference over the test articles before, it is rather surprising that author information alone can yield 0.8 in AUC score, even for very light authors who have received between 3 and 5 votes in total in the training data. This suggests that users' viewpoints are quite consistent: a large portion of the ratings can be adequately explained by the pair of user identities. One interesting observation is that the number of ratings required for **vv** to outperform **uc** in P@5 is quite high. This suggests that to obtain high precision at the top of a recommended list, comment features are important.

Nonetheless, modeling textual information in addition to author information provides additional improvements. Based on paired t-tests with signifi-



(a) AUC by author activity levels (b) AUC by rater activity levels (c) AUC by user activity levels

Figure 2: AUC of different models as a function of the activity level of authors or raters. The x-axis (bottom) has the form m-n, meaning the subset of the test data in which the number of ratings that each author received (as in (a)) or each rater gave (as in (b)) in the training set is between m and n. In (c), we select both authors and raters based on the m-n criterion. The x-axis (top) denotes the number of ratings in the subset

cance level 0.05, **vv+uc** significantly outperforms **vv** in all metrics if the author received < 500 ratings in the training set. Except for the very heavy authors, even for cases where both raters and authors are heavy users (Figure 2(c)), adding the comment feature information still yields additional improvement over the already impressive performance of using **vv** alone. In spite of the simple representation we adopted for the textual information, the full model is still capable of accounting for part of the residual errors from **vv** model (that uses authorship information alone) by using comment features — what was actually written does matter.

Finally, if we breakdown the comparison between **vv+uc** and **uc** for different user activity levels, **vv+uc** significantly outperforms **uc** (with level 0.05) in all metrics if the author received at least 5 ratings in the training set.

4.3.3 Analysis of textual features

Recall that we limited the vocabulary size to the 10K most frequent terms for efficiency reasons. Is this limitation likely to affect our model performance significantly? We examined the effect of different numbers of features. In the following table, #features = n means that both $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are bags of n words[3]. Since the **vv** model does not utilize rater or comment features, we examine AUC of the **uc** model.

#features	1K	3K	5K	10K
AUC	0.7713	0.7855	0.7872	0.7876

As can be seen, the performance improvement is in the 4th decimal place when we increase from 5K features to 10K features. Thus, we do not further increase the number of features in our experiments.

Note that our full model does not require rater features and comment features to be in the same feature space. Each is projected into the hidden "viewpoint" space, via G and D, separately. For simplicity and easy comparison to other methods, we used all comments liked by a rater in the past to build the feature vector of the rater. But since the full model already has information of the textual content of comments from the comment features, and which comments were liked by the users from the ratings, rater features constructed this way do not provide any new information. Indeed, if we model $\boldsymbol{u}_i \sim N(1, \sigma_u^2)$, instead of $\boldsymbol{u}_i \sim N(G\boldsymbol{x}_i, \sigma_u^2)$, this omission of x_i does not hurt the performance of the model. In future work, other meta-information about the rater

[3]Note that we used n most useful features in each case.

can easily be incorporated into $\boldsymbol{x}_i$ to enrich rater representation.

Recall that comment features $\boldsymbol{x}_j$ were projected to comment factors $\boldsymbol{c}_j$ via D. We envisioned that the comment factors could be representing viewpoints. Does our model conform to this intuition? Let's consider the simplest case, where we restrict $\boldsymbol{u}_i$ and $\boldsymbol{c}_j$ to be one-dimensional vectors. In this case, each can be represented by scalars u_i and c_j. If u_i and c_j are of the same sign, then the rater is likely to like the comment. Words assigned high positive weights or low negative weights via D will have significant contributions to the overall sign of c_j. Now if we examine such words, will we see any meaningful differences in the underlying viewpoints of these two groups of words?

To address this question qualitatively, we manually sampled words with heavy weights, focusing on politics-related ones (so that viewpoints are likely to be polarized and easier to interpret). At one extreme, we observe words like *repukes, repugs*, which seemed to be derogatory mentions of Republications, and likely to represent an anti-Republication point of view. At the other end, we observe terms like *libtards, nobama, obummer*. While terms like *nobama* may appear to be typos at first sight, a quick search online reveals that these are at least intentional typos expressing anti-Obama sentiments, which clearly represents an opposite underlying perspective from terms like *repukes*.

These examples also illustrate the importance to learn directly from the data of interest to us. Such indicative words would never have appeared in more formal writings. While we do not have direct labels for perspectives, our model seems to be capturing the underlying perspectives (as much as a unigram-based model could) by learning from user preference labels across different users. This allows us to learn the text features most relevant to our dataset, which is particularly important given the time-sensitive and ever-evolving nature of news-related comments.

5 Conclusions

In this paper, we promote personalized recommendation as a novel way of helping users to consume large quantities of subjective information. We propose using a principled way of incorporating both rater-comment and rater-author interactions simultaneously. Our full model significantly outperforms strong baseline methods, as well as previous methods proposed for text recommendation. In particular, learning weights over textual features across all users outperforms learning for each user individually, which holds true even for heavy raters. Furthermore, while using authorship information alone provides stronger signal than using textual information alone, to our surprise, even for heavy users, adding textual information yields additional improvements.

It is difficult to comprehensively capture user affinity to comments using a finite number of ratings observed during a certain time interval. News and comments on news articles are dynamic in nature, novel aspects may emerge over time. To capture such dynamic behavior, comment factors have to be allowed to evolve over time and such an evolution would also necessitate the re-estimation of user factors. Incorporating such temporal dynamics into our modeling framework is a challenging research problem and requires significant elaboration of our current approach.

Acknowledgments

We thank the anonymous reviewers for useful suggestions.

References

D. Agarwal and B.C. Chen. Regression-based latent factor models. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 19–28. ACM, 2009.

Jae-Wook Ahn, Peter Brusilovsky, Jonathan Grady, Daqing He, and Sue Y. Syn. Open user profiles for adaptive news systems: help or harm? In *Proceedings of the 16th international conference on World Wide Web (WWW)*, 2007.

Robert Bell, Yehuda Koren, and Chris Volinsky. Modeling relationships at multiple scales to improve accuracy of large recommender systems. In *KDD*, 2007.

D. Billsus and M. Pazanni. Adaptive news access. Springer, Berlin, 2007.

J.G. Booth and J.P Hobert. Maximizing generalized linear mixed model likelihoods with an automated

monte carlo EM algorithm. *J.R.Statist. Soc. B*, 1999.

Wei Chu and Seung T. Park. Personalized recommendation on dynamic content using predictive bilinear models. In *Proceedings of the 18th international conference on World Wide Web (WWW)*, 2009.

Cristian Danescu-Niculescu-Mizil, Gueorgi Kossinets, Jon Kleinberg, and Lillian Lee. How opinions are received by online communities: A case study on Amazon.com helpfulness votes. In *Proceedings of WWW*, pages 141–150, 2009.

Abhinandan S. Das, Mayur Datar, Ashutosh Garg, and Shyam Rajaram. Google news personalization: scalable online collaborative filtering. In *Proceedings of the 16th international conference on World Wide Web (WWW)*, 2007.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39:1–38, 1977.

Peter D. Hoff. Bilinear mixed-effects models for dyadic data. *Journal of the American Statistical Association*, 100(469):286–295, 2005.

Minqing Hu and Bing Liu. Mining and summarizing customer reviews. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, pages 168–177, 2004.

Daisuke Kawahara, Kentaro Inui, and Sadao Kurohashi. Identifying contradictory and contrastive relations between statements to outline web information on a given topic. In *Proceedings of the 23rd International Conference on Computational Linguistics (COLING): Posters*, 2010.

Soo-Min Kim, Patrick Pantel, Tim Chklovski, and Marco Pennacchiotti. Automatically assessing review helpfulness. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 423–430, Sydney, Australia, July 2006. Association for Computational Linguistics.

Y. Koren. Factor in the neighbors: Scalable and accurate collaborative filtering. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 4 (1):1–24, 2010. ISSN 1556-4681.

Michael Laver, Kenneth Benoit, and John Garry. Extracting policy positions from political texts using words as data. *American Political Science Review*, 97(2):311–331, 2003.

Wei-Hao Lin and Alexander Hauptmann. Are these documents written from different perspectives? A test of different perspectives based on sta tistical distribution divergence. In *Proceedings of the International Conference on Computational Linguistics (COLING)/Proceedings of the Association for Computational Linguistics (ACL)*, pages 1057–1064, Sydney, Australia, July 2006. Association for Computational Linguistics.

Wei-Hao Lin, Theresa Wilson, Janyce Wiebe, and Alexander Hauptmann. Which side are you on? identifying perspectives at the document and sentence levels. In *Proceedings of the Conference on Natural Language Learning (CoNLL)*, 2006.

Jingjing Liu, Yunbo Cao, Chin-Yew Lin, Yalou Huang, and Ming Zhou. Low-quality product review detection in opinion summarization. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 334–342, 2007. Poster paper.

Yue Lu, Panayiotis Tsaparas, Alexandros Ntoulas, and Livia Polanyi. Exploiting social context for review quality prediction. In *Proceedings of the 19th International World Wide Web Conference (WWW)*, 2010.

P. McCullagh and J. A. Nelder. *Generalized Linear Models*. Chapman & Hall/CRC, 1989.

Tony Mullen and Robert Malouf. A preliminary investigation into sentiment analysis of informal political discourse. In *AAAI Symposium on Computational Approaches to Analysing Weblogs (AAAI-CAAW)*, pages 159–162, 2006.

Tony Mullen and Robert Malouf. Taking sides: User classification for informal online political discourse. *Internet Research*, 18:177–190, 2008.

Bo Pang and Lillian Lee. Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, 2(1-2):1–135, 2008.

Ana-Maria Popescu and Oren Etzioni. Extracting product features and opinions from reviews. In *Proceedings of the Human Language Technology Conference and the Conference on Empirical Methods in Natural Language Pro cessing (HLT/EMNLP)*, 2005.

Steffen Rendle and Schmidt-Thie Lars. Pairwise interaction tensor factorization for personalized tag recommendation. In *Proceedings of the third ACM international conference on Web search and data mining*, WSDM '10, pages 81–90, New York, NY, USA, 2010. ACM.

R. Salakhutdinov and A. Mnih. Bayesian probabilistic matrix factorization using Markov chain Monte Carlo. In *Proceedings of the 25th international conference on Machine learning*, pages 880–887. ACM, 2008a.

R. Salakhutdinov and A. Mnih. Probabilistic matrix factorization. *Advances in neural information processing systems*, 20:1257–1264, 2008b.

Badrul Sarwar, George Karypis, Joseph Konstan, and John Reidl. Item-based collaborative filtering recommendation algorithms. In *WWW '01: Proceedings of the 10th international conference on World Wide Web*, pages 285–295. ACM, 2001.

Benjamin Snyder and Regina Barzilay. Multiple aspect ranking using the Good Grief algorithm. In *Proceedings of the Joint Human Language Technology/North American Chapter of the ACL Conference (HLT- NAACL)*, pages 300–307, 2007.

Swapna Somasundaran and Janyce Wiebe. Recognizing stances in online debates. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, 2009.

Swapna Somasundaran and Janyce Wiebe. Recognizing stances in ideological on-line debates. In *Proceedings of the NAACL HLT 2010 Workshop on Computational Approaches to Analysis and Generation of Emotion in Text*, 2010.

David H. Stern, Ralf Herbrich, and Thore Graepel. Matchbox: large scale online bayesian recommendations. In *Proceedings of the 18th international conference on World Wide Web (WWW)*, 2009.

Matt Thomas, Bo Pang, and Lillian Lee. Get out the vote: Determining support or opposition from Congressional floor-debate transcripts. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 327–335, 2006.

Ivan Titov and Ryan McDonald. A joint model of text and aspect ratings for sentiment summarization. In *Proceedings of the Association for Computational Linguistics (ACL)*, 2008.

Jun Wang, Arjen P. de Vries, and Marcel J. T. Reinders. Unifying user-based and item-based collaborative filtering approaches by similarity fusion. In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 501–508, New York, NY, USA, 2006. ACM.